Université Paris-Dauphine
Ecole Doctorale de Gestion de l'Université Paris-Dauphine (EDOGEST)
Dauphine Recherches en Management (DRM)



L'organisation de la résolution de problème dans les communautés Open Source

---

Thèse pour l'obtention du titre de docteur en sciences de gestion
(Arrêté du 7 Août 2006)

Présentée et soutenue publiquement par

Héla MASMOUDI

Jury :

Directeur de thèse : Jean-Michel DALLE, Professeur à l'Université de Pierre et Marie Curie

Imed Boughzala, Maitre de conférences HDR à

Rapporteurs : Pierre-Jean BENGHOZI, Professeur à l'Ecole Polytechnique

Suffragants: Jean-charles POMEROL, Professeur à l'université de Pierre et marie curie

Michel POIX, Maitre de conférences HDR à l'université Paris-Dauphine

«L'Université Paris Dauphine n'entend donner aucune approbation ou improbation aux opinions émises dans les thèses. Ces opinions doivent être considérées comme propres à l'auteur.»

A mes parents, à ma famille, et à mes amis.

Table des matières

Introduction générale

Dans un environnement hautement dynamique et complexe, les entreprises se retrouvent à la recherche sans cesse de sources potentielles de valeur et de performance, reposant de plus en plus sur la mise en commun de ressources pour faire face à cette complexité grandissante.

Dans un ouvrage intitulé *Distributed Work*, Hinds et Kisler (2002) expliquent, dans sa première partie, que la performance d'une organisation dépend de la diversité des ressources qu'elle déploie et que cette diversité est réalisée par la distribution du travail. L'intérêt accordé à cette notion s'est accru avec la généralisation des nouvelles technologies de l'information et de la communication NTIC, et particulièrement d'Internet et du Web.

La distribution fait donc référence à la dispersion géographique des acteurs, et c'est à travers l'usage des NTIC et des novelles formes de communications, que les limites de l'activité collaborative sont étendu et que les ressources de l'entreprise en connaissances s'amplifient (King et Frost, 2002). Au même temps, la notion de distribution signifie l'ouverture, des compétences diverses venant d'univers différents (Benkler, 2002 ; von Hippel, 2005), et un travail réparti (Moon et Sproull, 2002).

C'est dans ce contexte qu'apparaissent les communautés Open Source, proposant un espace de travail distribué, alliant à la fois éloignement géographique, ouverture, diversité et répartition.

Les produits résultants, les logiciels, revendiquent aujourd'hui, une qualité intrinsèquement supérieure: cette meilleure qualité intrinsèque serait liée au fait que ces produits sont créés sur Internet par des communautés de développeurs indépendants, pour la plupart eux-mêmes utilisateurs (Gensollen, 2004). Il est un fait que ces communautés peuvent promouvoir la créativité et l'innovation, en favorisant à la fois l'accumulation, la génération et la distribution des connaissances (Foray, 2004; Cohendet et al., 2003).

I. L′Open Source : une nécessite pour toute entreprise

Récemment, dans l'Open Word Forum 2011(OWF)[1], un forum rassemblant des experts de l'Open Source, les usagers, mais également des acteurs institutionnels européens et français, l'accent été mise sur les enjeux l'Open Source non seulement pour les entreprises mais également pour les services public notamment avec l'introduction du concept de l'*Open data* s'inscrivant dans le thème de la transparence des données publiques (Nigel Shadbolt, 2011).

« Aujourd'hui plus que 85%[2] *des entreprises utilisent déjà des logiciels Open Source »*, souligne Patrice Bertrand (2011) directeur général de Smile, une société d'ingénieurs experts dans la mise en œuvre de solutions Open Source.

D'après une étude réalisée en 2010 par Forrester Research[3], l'ampleur qu'occupe l'Open source dans le monde des entreprises s'explique par les bénéfices qu'accordent les logiciels Open Source en terme de budgets, mais principalement en terme de liberté de choix, d'ouverture, de dynamique de développement et d'indépendance dans les choix.

Les entreprises sont ainsi de plus en plus à la recherche d'espaces, favorables aux échanges et à l'innovation et ceci est rendu possible par les propriétés qu'accorde l'Open Source en terme d'ouverture, de liberté, de diversité, etc.

Cet enjeu pousse aujourd'hui de nombreuses entreprises de vouloir maîtriser l'insertion de l'Open Source dans le système d'information, pour en tirer le meilleur bénéfice. Et la première étape d'une politique maîtrisée est souvent de faire un état des lieux de ce qui est déjà déployé. Notre apport consiste alors à dresser cet état de lieu en nous centrant sur les aspects organisationnels qui distinguent les communautés Open Source.

---

[1] L'Open World Forum (OWF) 2011 s'est tenu le 22, 23 et 24 septembre à Paris. Cet événement annuel a été lancé en 2008, et se tient désormais chaque année à Paris. Cette édition a notamment pu compter sur la présence d'Eric Besson, ministre de l'industrie, de l'énergie et de l'économie numérique, Jean-Paul Planchou, vice-président de la région Ile-de-France en charge du développement économique, de l'innovation, et de la technologie, Jean-Louis Missika, adjoint au maire de Paris chargé de l'Innovation, la recherche et les universités.

[2] Ce chiffre se base sur une étude élaborée par Gartner en 2009, une entreprise américaine de conseil et de recherche dans le domaine des techniques avancées, crée en 1979.

[3] Forrester Research est une entreprise spécialisée des études de marchés et s'intéresse particulièrement à l'impact des technologies dans le monde des affaires.

II. Emergence des questions de recherches et définition des objectifs

Depuis le début du XXI ème siècle, l'Open Source est devenu un élément incontournable. De nombreux chercheurs et équipes de recherche (notamment Lerner et Tirole, 2002 ; Lakhani et von Hippel, 2003 ; ou encore von Krogh et von Hippel, 2006), des revues scientifiques (voir par exemple les numéros spéciaux de *Research Policy* de 2003, de *Management Science* en 2006 et plus récemment d' *Information Economics and Policy* de 2008) ont investi cet objet d'étude tant du point de vue de sa réussite commerciale, de ses conséquences sur les systèmes d'information que de ses fondements organisationnels.

Dans cette thèse, c'est la nature des organisations qui produisent les logiciels Open Source que nous allons interroger, du point de vue des mécanismes qu'elles mettent en place précisément pour trouver des solutions aux problèmes qui se posent à elles. De ce point de vue, l'ouverture du code des logiciels libres, et le plus spécifiquement la distribution des déclarations de bogues, permet à un plus grand nombre de développeurs de proposer des solutions et de résoudre par conséquent plus rapidement et plus efficacement les problèmes. Mais est-ce vraiment le cas dans la réalité? Plus généralement, y aurait-il des modes spécifiques qui caractériseraient toutes les communautés de logiciels libres, ou au moins certaines?

L'examen de la littérature sur les questions relatives à l'organisation des communautés Open Source montre que certes, l'impression d'anarchie qui peut se dégager de ces communautés, introduite par le théoricien du mouvement Open Source Eric Raymond (1999), est largement faussée, et que de nombreux travaux ont montré qu'au contraire ces communautés sont souvent très hiérarchiques et centralisées (Crowston et Howison, 2006) – au premier rang desquelles la communauté du système d'exploitation Linux (Lee et Cole, 2003)– et que le leadership existent dans la plus part des projets Open Source (Mockus et al., 2002). Mais cela ne résout rien du point de vue vis-à-vis de la question qui nous concerne, car ces communautés élaborent alors dans ce contexte un certain nombre de processus organisationnels "communautaires" différents de ceux des organisations traditionnelles.

En effet, plusieurs chercheurs ont tentés de mettre en évidence la dimension hiérarchique qui spécifie les communautés Open Source. Dans une étude du noyau Linux, Lee et Cole (2003) montrent que la hiérarchie est reflétée à travers une distribution inégale du pouvoir

entre le noyau central du projet et les autres participants. D'après Mockus et al. (2002) c'est à travers les actions que les différents niveaux hiérarchiques d'une communauté Open Source sont identifiés. Leur étude des projets Mozilla et Apache indique que les tâches d'intégrations, de validation, de correction et de déclarations sont réparties entre les participants, et que la complexité des tâches définis l'ordre des niveaux hiérarchiques. Crowston et Scozzi (2005) proposent quand à eux une description générale de l'organisation sociale des projets Open Source en élaborant le modèle de l'oignon, selon lequel, les projets Open Source sont organisés autour d'une équipe centrale (core team) composé d'un nombre limité de développeurs compétent qui décident des tâches les plus critique de projet (tâches de conception), tant dis que les tâches les moins critiques sont généralement déléguées à une partie plus large, la périphérie, qui a moins de pouvoir décisionnel.

L'examen de ces recherches montre bien que l'organisation des communautés Open Source repose sur une structure hiérarchie. Néanmoins, la nature des mécanismes mis en œuvre et qui régissent l'organisation et coordination au sein de ces communautés n'est pas encore vraiment élucidée. Nous pensons que cette limite appelle à l'approfondissement des enquêtes empiriques sur les questions relatives à l'organisation des communautés Open Source en utilisant une approche centrée sur la communication entre les participants.

III. L'analyse de la communication pour étudier l'organisation des communautés Open Source

L'étude de l'interaction ou de la communication tels que nous l'adoptons dans les communautés Open Source a fait l'objet d'un nombre d'étude. Ducheneaut (2005) a mis en évidence l'évolution des développeurs, d'une position périphérique à une position centrale dans le réseau d'acteurs du projet Phython. Mockus, Fielding et Herbsleb (2002) tentent également à travers l'étude des interactions, de comparer le processus d'intégration des participants dans deux projets Mozilla et Apatche. C'est les travaux de Sandysky, Gasser et Ripoche (2004 ; 2005; 2006) qui se rapprochent plus à notre étude. D'abord, nous analysons le même cas d'étude, à savoir la communauté Mozilla en nous centrons sur les échanges en rapport avec la résolution de bogues au sein de Bugzilla. Ensuite, c'est à travers l'analyse des interactions ou la communication, que nous étudions l'organisation des communautés Open Source.

Dans cette thèse, nous adaptons une approche différente :

- Premièrement, notre approche vise à analyser la communication entre participants dans le but de rendre compte de l'activité des participants, en interactions les un avec les autres et avec les artefacts technologiques, et identifier par la suite les fonctions de la communication dans l'organisation de l'activité. La communication englobe donc à la fois les activités liées à la définition des problèmes et à l'élaboration de solutions (par exemple, la description, les directives, la conception de solution) et les activités interactionnelles (interaction directe, dialogue, relance, etc.).

- Deuxièmement, notre approche vise à comparer les statuts des participants et les actions, l'analyse permet de comprendre les rôles des participants dans l'organisation de l'activité en fonction de leurs positions hiérarchiques dans la communauté.
- Enfin, notre approche vise à identifier sur la base de la variation des différentes propriétés organisationnelles identifiées (genre linguistique (catégories), statuts des participants, statut du problèmes, etc.), de déterminer différents modes de coordination.

IV. La démarche de la recherche

Telle qu'illustré dans la figure 0.1, nos questions de recherche émergent de l'association des éléments de la première revue de littérature sur les communautés Open Source et des premiers résultats de recherches. Notre démarche de recherche se base donc sur une démarche abductive reposant à la fois sur les réalités tirées du terrain, mais également sur les concepts théoriques déjà élaborées. Le choix de cette démarche est lié d'une part au fait que l'approche que nous adoptons pour étudier l'organisation des communautés Open Source n'a pas été abordée par les travaux académiques. Une première incursion sur le terrain nous permet donc d'avoir une vision plus claire sur la réalité. D'autre part, la richesse de la littérature scientifique représente un cadre de référence sur lequel nous nous appuyons pour donner du sens aux observations et développer nos réflexions.

De point de vue méthodologique, le plan d'étude que nous avons choisit est multi méthode puisque nos analyses se base à la fois sur approche qualitative et quantitative. Notre étude est composée de trois phases. La première phase est exploratoire et la méthode d'analyse

utilisée dans ce cadre est qualitative. Les deuxième et troisième phases sont constructives. Pour ces deux dernières phases, nous nous sommes basé sur les deux démarches quantitatives, puis qualitatives.

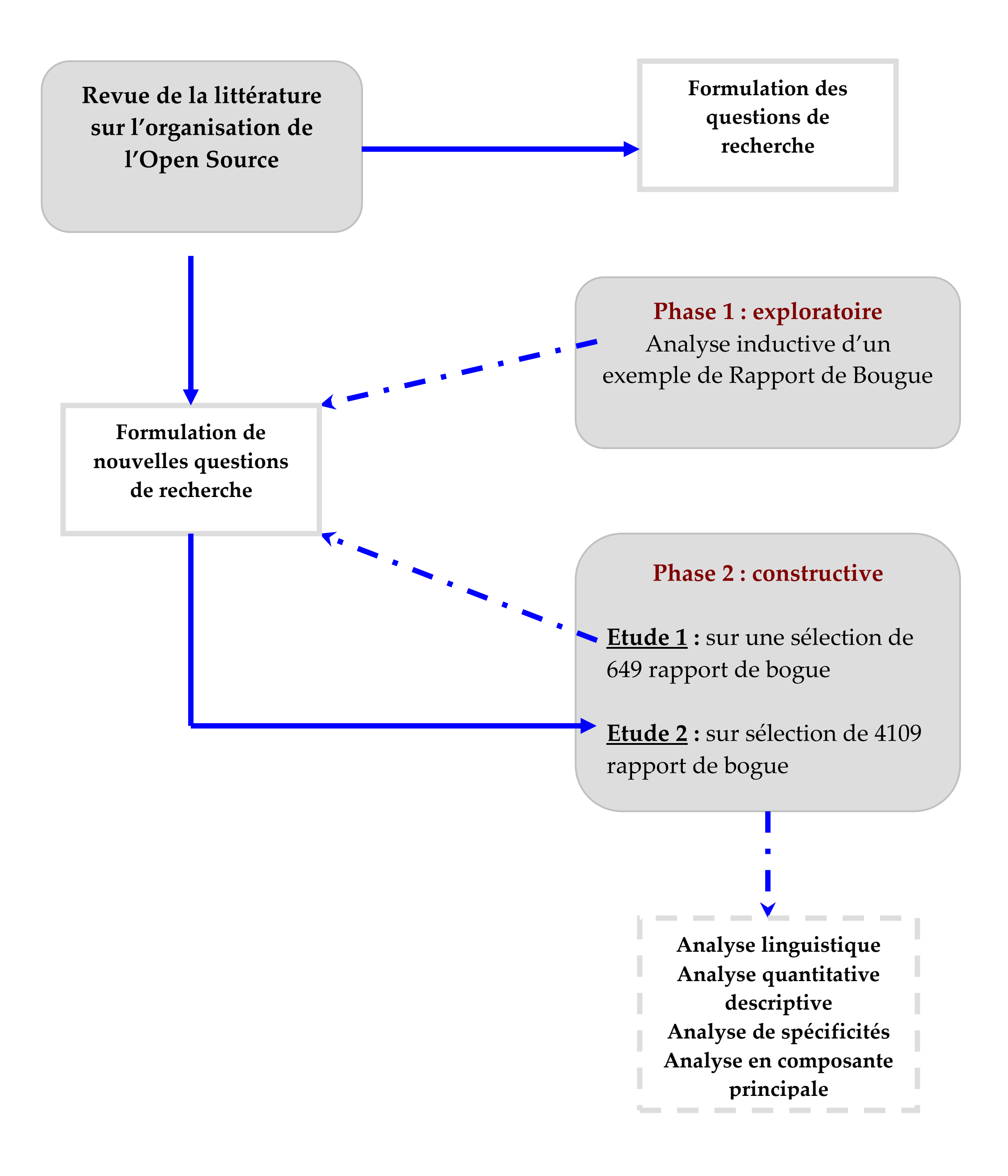
La phase exploratoire a permis, à travers l'étude d'un exemple de rapport de bogue, de comprendre le déroulement de l'activité de la résolution de problème au sein de Bugzilla et d'identifier certains aspects explicitant l'organisation de cette activité. Pour analyser ce rapport de bogue nous avons utilisé la méthode développée par Strauss et Corbin (1990), une méthode qui consiste à étudier les échanges entre les développeurs en les examinant ligne par ligne tout en faisons le lien avec l'évolution de l'activité de résolution.

Les deux phases constructives, quand à elle, ont permis de répondre aux différentes questions de recherches issues des interactions entre la revue de la littérature et les différents constats du terrain de recherche. Pour répondre à ces questions, deux corpus ont été constitués.

Dans premier temps une analyse quantitative de 649 rapports de bogues à permis de déterminer, à travers une méthode de l'analyse linguistique basée sur les genres, le rôle de la communication dans l'organisation de l'activité au sein de Bugzilla. A l'issu de cette seconde étape nous avons identifié quatre catégories linguistiques caractérisant la communication partir desquelles nous avons plu définir les fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla.

Cette construction, à servis dans un second temps comme base d'analyse et de questionnements. En effet, l'analyse des résultats de recherches obtenus et la revue de la littérature, nous ont permis d'avancer de nouvelles questions de recherche. Pour réponde à ces questions nous avons étudier un corpus composé de 4109 rapports de Bogues critiques, nous avons recouru à différentes techniques d'analyse, à savoir, des analyses des spécificités linguistqies, des analyses descriptives, et des analyses en composante principale (ACP).

Figure 0.1 : démarche générale de la recherche



V. Plan de la thèse

Ce travail de recherche s'articule en six chapitres.

Le chapitre I présente le contexte général de l'étude qui est la résolution de problèmes distribués. Il met en évidence l'évolution des concepts de la résolution de problème abordés dans la litérature et met l'accent sur le rôle de la distribution dans des contextes complexes imposant l'élargissement de l'espace la résolution par une ouverture à des connaissances et des compétences externes. Le développement Open Source est par la suite présenté comme étant un modèle de résolution distribuée, en s'appyuant sur le cas du peojet Mozilla.

Le chapitre II est consacré à la présentation des fondements théoriques sur l'organisation des communautés Open Source. Nous nous intéréssons dans un premier temps aux concepts de la communauté en générale et de la communauté Open Source en particulier. Plusieurs définitions de ces concpets sont présentées. Les caractéristiques dégagées de ces définitions nous conduit dans un second temps à se à se pencher sur les particularités de ces communautés à travers l'analyse de sa structure organisationelle.

Le chapitre III traite la coordination dans les communautés Open Soure. Nous présentons en première partie les définitions de la coordination selons les différentes approches abordées dans la littérature. Ensuite nous distinguons, sur la base des traveaux présentés jusqu'as présent, les différentes formes de coordinations propres aux communautés de manière générale et aux communautés Open Source de manière spécifique.

Le chapitre IV présente notre démarche globale de recherche et décrit les différentes étapes poursuivies pour confronter les connaissances théoriques avec notre démarche empirique. Il expose notre positionnement épistémologique et nos choix méthodologiques.

Le chapitre V expose les résultats obtenus à l'issu de nos deux premières phases empiriques. Premièrement, il présente les résulats de la phase exploratoire. Il décrit le contexte de la résolution de problème dans le cas que nous étudions Bugzilla, sur la base de l'analyse exploratoire inductive d'un exemple de rapport de bogue. Deuxièmement, il discute les résultats obtenus suite à l'analyse linguistique du premier corpus. Nous répondons ainsi aux premières interrogations en définissant quatres catégories linguistiques spécifiques à Bugzilla. Quatre fonctions de la communication dans l'organisation de la résolution de bogues sont par la suite identifiées.

Le chapitre VI présente les résultats de la troisième phase empirique. Dans premier temps, il identifie les rôles des participants en fonctions de leurs status dans le projet (cœur ou périphérie). Il définit dans un second temps trois modes de coordination spécifique au contexte de Bugzilla.

La conclusion générale offre une synthèse des approts théoriques et des implications managériales de la recherche. Les résulats de notre recherche nous conduisent donc à proposer une vision renouvlé de l'Organisation des communauté Open Soure, selon une approche centrée sur la communication et l'interaction entre les contributeurs.

PREMIRE PARTIE :
PRESENTATIONS DU CADRE THEORIQUE ET METHODOLOGIQUE DE LA RECHERCHE

PREMIERE PARTIE : PRESENTATION DU CADRE THEORIQUE ET METHODOLOGIQUE DE LA RECHERCHE

CHAPITRE I-LA RESOLUTION DE PROBLEMES DISTRIBUES DANS LE CADRE DU DEVELOPPEMENT OPEN SOURCE : LE CAS DE MOZILLA

Section 1.*Vers une nouvelle conception de la résolution de problème : la distribution*
Section 2. *Développement Open Source et résolution de problème distribuée, le cas du projet Mozilla*
Section 3. *Synthèse et apport a notre questionnement*

CHAPITRE II- STRUCTURATION ET ORGANISATION DE L'OPEN SOURCE

Section 1. *Les communautés Open Source : Apports et limites de la littérature sur leurs caractérisations*
Section 2. *L'apport de la littérature aux mécanismes de régulation des communautés Open Source*
Section 3. *Synthèse et apport a notre questionnement*

CHAPITRE III-LA COORDINATION DANS LES COMMUNAUTES OPEN SOURCE

Section 1. *Les différentes approches de la coordination*
Section 2. *La coordination au sein des communautés Open Source*
Section 3. *Synthèse et Apport a notre questionnement*

CHAPITRE IV- FONDENMENTS EPISTEMOLOGIQUES ET CHOIX METHODOLOGIQUES

Section 1. *Le Positionnement épistémologique*
Section 2. *L'objet de notre recherche*
Section 3. *Le Choix méthodologiques : un plan d'étude multi-méthode*
Section 4. *Conclusion et synthèse sur les différentes étapes de la recherche*

CHAPITRE I - LA RÉSOLUTION DE PROBLEMES DISTRIBUES DANS LE CADRE DU DEVELOPPEMENT OPEN SOURCE : LE CAS MOZILLA

TABLE DES MATIERES

CHAPITRE I - LA RÉSOLUTION DE PROBLEMES DISTRIBUES DANS LE CADRE DU DEVELOPPEMENT SOURCE : LE CAS MOZILLA

« *Given enough eyeballs, all bugs are shallow* »
(Raymond, 2001)

Le but de ce premier chapitre est de présenter le contexte général de l'étude, en développant le concept de la résolution de problèmes distribués. Un retour historique sur les principales approches de la résolution de problèmes nous permet d'avoir une compréhension des concepts développés dans ce domaine ainsi que de leurs limites.
Nous traitons dans une deuxième partie, l'approche de la résolution de problèmes distribués. La résolution de problèmes dans le cadre du développement Open Source est ensuite présentée en tant que forme prototypique de la résolution de problème distribué. Plus particulièrement, nous nous intéressons au projet Mozilla qui nous sert d'objet d'analyses et qui illustrent une véritable activité de résolution de problème distribué.
Dans une dernière section, une synthèse des principaux éléments de ce chapitre est présentée et nous permettra de mettre en avant les apports de cet éclairage théorique pour notre objet de recherche.

Section 1. Vers une nouvelle conception de la résolution de problème : la distribution

Traditionnellement, les travaux traitent les problèmes comme étant différents, admettant plusieurs solutions, mais présentant néanmoins un schéma de résolution commun. L'objectif été de définir une méthode générale de résolution permettant le traitement de catégories de problèmes très larges. Dans ce cadre, les études de Newel et Simon (1972-1979) occupent une place importante tant en psychologie qu'en intelligence artificielle. Ces études ont jouées un rôle fondamental dans l'élaboration de procédure de résolution.

Cependant, la résolution de problèmes n'est pas une activité uniforme. Jonassen (2000) montre qu'il n y a pas une unique trajectoire de résolution, et que cette dernière varie selon plusieurs facteurs tels que la nature du problème ou bien les compétences des individus qui participent à la résolution. Des problèmes complexes, mal structurés ou encore mal définis ne sont donc pas résolus de la même manière. De même, ces problèmes impliquent des personnes ayant des niveaux de compétences très variées, qui ont l'habitude ou pas de travailler ensembles.

Smith (1991) regroupe ces facteurs en deux catégories : les facteurs externes et internes. Les premiers représentent l'ensemble des facteurs qui prennent en considération la variation des types de problèmes. Les facteurs internes quant a eux décrivent les variations dans la résolution de problèmes, et concernent, surtout, les différences individuelles.

Parallèlement, Davis et Smith (1981) présentent un nouveau concept de la résolution de problèmes qui tient en compte leurs complexités. Ce nouveau concept est la résolution de problèmes distribués (*Distributed problem solving*). Selon les auteurs, la complexité d'un problème nécessite qu'ils soient résolus de manière collaborative et distribuée.

Plus tard, Lakhani et Von Hippel (2005) développent un concept similaire: la recherche diffusée (*Broadcast research*). Ils montrent que la diffusion du problème permet d'élargir le champ du problème en ayant accès à des compétences et des expertises externes. D'après Lakhani (2005), le développement des outils de communications font émerger des problèmes plus complexes, mais également de nouveaux espaces de communications très riches en connaissances. Diffuser le problème dans de tels espaces présente l'avantage de trouver rapidement des solutions.

Nous nous attachons, dans cette partie, à présenter le concept de la résolution de problèmes distribués. Nous reviendrons, dans un premier temps, sur les études de Newell et Simon (1972/1979) des heuristiques de résolution et leurs rôles fondamentaux dans l'élaboration de procédures de résolution. Nous exposons, dans un second temps, le concept de la résolution de problèmes distribués.

I.A. Les approches classiques de la résolution de problèmes : heuristiques et procédures de résolutions (Newell et Simon, 1972/1979)

Nous présentons dans cette partie deux axes de réflexion de Newell et Simon (1972/1979). Dans un premier axe, les auteurs (1972) décrivent la résolution de problème comme une activité cognitive, et développent un modèle permettant de représenter cette activité. En partant de ce modèle, Newell et Simon (1979) proposent de construire un outil qui permet de reproduire les mécanismes de prise de décision humaine, par la création de programme. Cet outil est l'intelligence artificielle.

I.A.1. Les apports des sciences cognitives

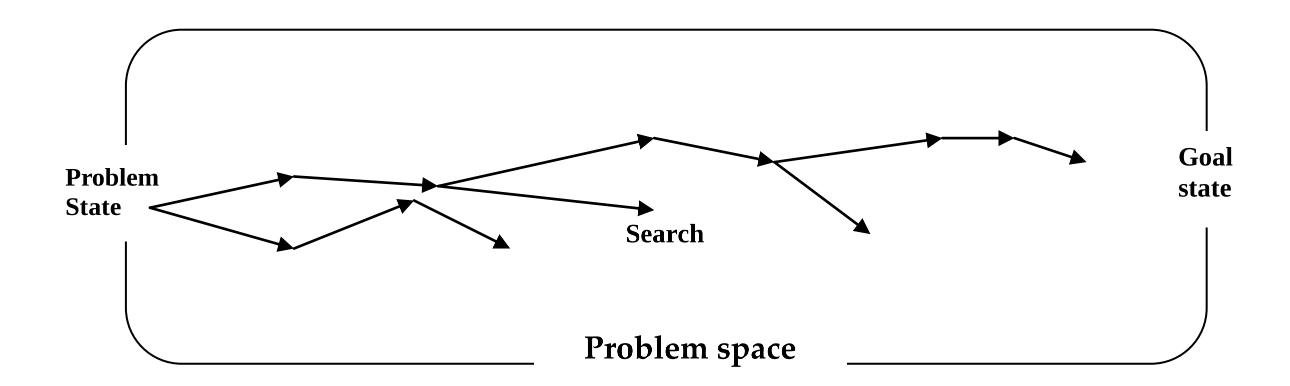
Pendant de nombreuses années, Newell et Simon (1972/1979) représente la référence dans les recherches en sciences cognitives et particulièrement dans les recherches qui caractérisent la résolution de problème comme une activité cognitive, notamment à travers ses travaux intitulés « *The structure of illstructured problems* » (1973) et « *Human Problem Solving* » (1972).

En effet, et sur la base de nombreuses études expérimentales, Newell et Simon (1972) développent un modèle qui caractérise tout les problèmes que l'on peu rencontrer et que l'on cherche à résoudre. Ce modèle, présentée dans la figure ci–dessous, bien qu'ancien, fourni le schéma de base pour les recherches dans le domaine de la résolution de problème. Selon Newell et Simon (1972), tous les problèmes apparaissent dans un « *espace problème* » et la résolution d'un problème met en œuvre des méthodes de recherche au sein de cet espace. Ce dernier est constitué de trois éléments :

- *l'état de problème* : il est présenté dans le modèle comme un point permettant de décrire le problème et constitue le point de départ du processus de résolution ;
- *l'espace de recherche* : il est présenté, comme l'indique la figure 1.1, par des lignes allant de l'état de problème au but visé. C'est l'espace de connaissances nécessaire à l'élaboration de procédures pour la résolution de problèmes (l'objectif final) ;
- *l'objectif final* : il représente la dernière étape de la résolution du problème. Il est présenté dans la figure suivante par un point, ce qui indique qu'il existe une seule réponse au problème.



Figure 1.1. Conception de l'espace problème (Newell et Simon, 1972)



Newell et Simon (1972) distinguent plusieurs méthodes de résolution. La méthode la plus simple consiste à chercher par « *essai-erreur* » une solution dans l'espace problème. Il s'agit de chercher des solutions parmi celles pré-existantes dans l'espace problème, d'appliquer et d'évaluer par la suite le résultat de chaque application jusqu'à la résolution définitive du problème. Selon Simon (1992), chercher une solution parmi plusieurs possibilités n'est pas guidé par des procédures systématiques, mais par des « *heuristiques* » :

> « *Problem solving searches are selective in that they generally explore only a miniscule fraction of the total (and usually immense) number of possibilities. In most cases of interest, the selection of the paths to be searched is not governed by foolproof, systematic procedures, but by rules of thumb we call heuristics.* » (Simon, 1992, p. 65)

Pour Newell et Simon (1972) une heuristique de recherche constitue un opérateur, un ensemble de règles, susceptible de fournir une solution optimale au problème. Simon (1992) définit un opérateur comme un processus qui contribue à changer une situation donnée :

> « *By 'operator' is meant some process that will change the present situation*.» (Simon, 1992, p 67)

Les auteurs identifient plusieurs heuristiques de résolution de problèmes : la résolution par « *la réduction de l'écart au but* », par « *l'analyse moyen- fin* », par « *l'analyse*

régressive » et par « *l'analogie* »[4]. Chaque heuristique implique un processus de recherche différent. Dans son livre « *Economics, Bounded Rationalitu and the Cognitive Revolution* », Simon (1992) explique que l'analyse moyens-fins est l'une des heuristiques de base dans la résolution de problème. Elle est d'ailleurs au cœur du modèle général de problème proposé (GPS) par Newell et Simon (1972), que nous présentons dans la partie suivante.

Selon Simon (1992, p. 67), l'application de la heuristique analyse moyens-fins suppose que:

- une situation donnée « *un état actuel* » soit comparée avec une situation voulue « *un état but* », un écart entre ces deux état est donc mis en évidence (par exemple : j'ai une planche de longueur 5 mètres ; je veux une planche de longueur 3 mètres ; il y a donc une différence entre la longueur de la blanche et ce que je souhaite avoir comme longueur);
- on cherche à réduire l'écart existant entre l'état actuel (avoir une planche de longueur 5 mètres) et l'état voulu (avoir une planche de longueur 3 mètres). Pour se faire il faut trouver un opérateur (ou des opérateurs) qui permettent de réduire cet écart (par exemple : scier, raboter, percer). Ces opérateurs sont associés à l'écart en fonction du résultat d'expériences qui montrent ils sont capables de réduire ou d'annuler l'écart en question (par exemple : scier change la longueur);
- on tente donc d'appliquer l'opérateur à la situation donnée. Des fois, on aperçoit que l'opérateur ne peut être appliqué sans avoir a modifier certains aspects de la situation (par exemple : une planche doit être bien maintenue pour être sciée), alors il faut poser de nouveau but (de type 1), puis chercher à réduire l'écart entre les

[4] Dans le texte nous détaillons seulement l'heuristique la plus soutenue par Simon (1992, p67), à savoir l'heuristique moyens-fins. Nous présentons toutefois, dans les présentes notes, une description brève des autres heuristiques : la résolution par la réduction de l'écart au but consiste à chercher parmi tous les opérateurs possibles celui qui produit l'état « ressemblant le plus » à l'état but (ou un état intermédiaire visé).

- la résolution par *la réduction de l'écart au but* consiste à chercher parmi tous les opérateurs possibles celui qui produit l'état « ressemblant le plus » à l'état but (ou un état intermédiaire visé) ;
- la résolution par *analyse régressive* part du but visé pour atteindre l'état courant 'état de problème ». Le problème est donc résolu lorsque l'application d'un opérateur ramène à l'état de départ ;
- la résolution par analogie se base sur l'utilisation de méthodes qui se sont avérées efficace dans la résolution de problèmes similaires en les appliquant au problème initial.

deux but (par exemple : maintenir et couper la planche) et arriver enfin à la situation voulue.

I.A.2. Les apports de l'intelligence artificielle

Après avoir discuté l'approche cognitive de la résolution de problème, et la notion d'heuristique, nous donnons dans cette partie un aperçu sur la modélisation de la résolution de problèmes sur la base de l'intelligence artificielle. Nous montrons que les méthodes et les techniques de l'intelligence artificielle fournissent les moyens pour produire des modèles de résolution de problèmes. Nous présentons enfin le modèle fondateur GPL créer par Newell et Simon (1972).

En intelligence artificielle, la pensée est considérée comme un processus analogue à celui d'un ordinateur. Pour Simon (1969/1996) il est donc possible de construire des programmes informatiques réalisant des tâches accomplies par des êtres humains, dans l'objectif de rendre plus performante la résolution de problème.

> « *Comme le cerveau, un ordinateur est capable de combiner des symboles. Ces symboles sont des formes physiques (telles que des traits de craie sur un tableau noir) qui peuvent se présenter comme des composants de structures (que l'on appelle parfois des expressions… En conséquence, rien n'empêche de faire raisonner un ordinateur comme un cerveau.* » (Simon, 1969 ; 1996, traduit par Le Moigne, 2004, p. 58).

Selon Simon (1969 ; 1996), l'intelligence artificielle fourni des programmes qui permet de décrire et d'expliquer la manière dont les individus comprennent et résolvent les problèmes:

> «*…les programmes fournissent un ensemble de mécanismes de base, une théorie, pour expliquer comment les êtres humains sont capable de comprendre les problème à la fois dans les nouveaux domaine auxquels ils ne connaissent rien et dans les domaines sur lesquels ils ont une plus ou moins grande quantité de connaissances antérieure*s.» (Simon, 1969 ; 1996, traduit par Le Moigne, 2004, p. 171).

Dans le livre « *Human Problem Solving* », Newell et Simon (1972) créent un modèle général de la capacité humaine à résoudre le problème appelé GPS « *Général Problem Solving* ». Le GPS, un modèle ancien datant de 1972, est le système de résolution de problème le plus renommée et utilisé comme références pour de nombreux travaux.

Pour créer ce modèle, Newell et Simon (1972) considèrent que l'efficacité dans la résolution de problèmes réside dans la capacité humaine à stocker et utiliser les connaissances. Le comportement humain est dans cette veine présenté comme une activité de résolution de problèmes. Cette activité se base sur un système de traitement d'information qui peut être formalisé par différents programmes[5]. Les auteurs considèrent également que le comportement humain dans la résolution de problème, ou le comportement du « *problem solver* » dépend fortement de l'environnement de la tâche « *task environment* ». En effet, comme nous avons défini précédemment, Newell et Simon (1972) représente une tâche à résoudre comme un espace de problèmes délimité, dans lequel le processus de résolution a lieu. Selon eux c'est l'environnement de la tâche « *task environnement* » qui détermine la structure de l'espace de problème :

> « *A man, viewed as a behaving system is quite simple. The apparent complexity of its behavior over time is largely a reflection of the complexity of the environment in which he finds himself.* » (Newell et Simon 1981, p. 65).

Les bases du modèle GPS se trouvent dans le mécanisme central de la résolution de problème, l'analyse moyens-fins « *means-endsanalysis* ». Comme expliqué précédemment, ce mécanisme définis un problème comme la différence entre deux états : un état courant A et un état souhaité B. GPS traite ce problème par un programme qui crée le but de transformer un objet de l'état A à l'état B. Ceci, en sélectionnant à chaque étape de la résolution, un opérateur qui permet de faire avancer la résolution de problèmes. Si le programme ne trouve pas de différence entre A et B, il applique des opérateurs O, O', O'', etc. aux objets A, A', A'' qui crée a chaque fois les objets A', A'', etc. jusqu'a avoir des attributs identiques à ceux de B pour s'arrêter avec succès.

Le modèle GPS à inspiré beaucoup d'autres chercheurs qui ont développés a leur tour des modèles représentant l'activité de résolution de problème. Nous citons, à titre d'exemple, le modèle IDEAL de Bransford et Stein (1984). Comme le modèle GPS, IDEAL décrit la résolution de problème comme un processus uniforme composé de plusieurs étapes : 1) identifier le problème, 2) définir et représenter le problème, 3) explorer les différente stratégies pour résoudre le problème, 4) mettre en application une stratégie, et 5) évaluer l'efficacité de la stratégie appliquée. Gick (1986) propose également un modèle plus

[5] Formaliser signifie écrire un système qui implémente l'ensemble du système de traitement de l'information. Un système est une collection de programmes d'ordinateur. Il existe des différents programmes puisque les problèmes peuvent être traité de différentes façons et sont de structures différentes.

simplifié, et présente le processus de résolution de problèmes comme étant composé de trois étapes : 1) *constructing problem representation*, 2) *searching for solutions*, et 3) *implementing and monitoring solutions*.
Ces formes de raisonnement ont jouées un rôle fondamental dans l'élaboration de procédure ou méthode générale de résolution. Dès lors, la résolution des problèmes analysés par Newell et Simon (1972), Bransford et Stein (1984), et Gick (1986) est conçue dans un espace limité. Dans cet espace les problèmes ont un seul point de départ, et un unique chemin de résolution.

I.B. Le concept de la résolution de problèmes distribués

Les conceptions traditionnelles de la résolution de problèmes, présentées dans la partie précédente, s'appuient fortement sur les expériences antérieures des auteurs (solvers). Les connaissances provenant de ces expériences représentent ainsi des points de départ pour la recherche d'une solution. Leurs arguments sont liés au fait que les auteurs ont tendance à utiliser les solutions et les méthodes servies pour la résolution de problèmes similaires. Selon Allen et Marquis (1964), ceci résulte d'une recherche locale de solutions. En effet, dans le cadre d'une recherche locale, la résolution se passe dans un espace problème. Dans ce cas, la recherche locale permet grâce à un effet d'apprentissage basé sur la répétition, de trouver rapidement une solution aux problèmes. (March et Simon, 1958 ; Nelson et Winter, 1982 ; Stuart et Podony, 1996).
Néanmoins, les problèmes que l'ont rencontre sont de plus en plus complexes et difficiles à résoudre par la simple exploration d'un espace. Aujourd'hui, au travers d'Internet et du Web, l'activité de résolution peu être menée collectivement et distribuée du point de vue géographique. De ce fait, les espaces de recherche sont devenus plus large, donnant accès à des connaissances diversifiées, et créant des espaces favorables au travail collectif et à la résolution de problèmes que nous sommes incapables de résoudre individuellement.

I.B.1. Propriétés du travail distribué

C'est depuis l'apparition et la généralisation des NTIC, et particulièrement d'Internet et du Web, comme infrastructure incontournable, que l'activité collective est rendue possible et que des chercheurs ont commencé à se pencher sur les questions liées au travail distribué. Bien que la notion de travail distribuée ne soit pas nouvelle, elle suscite aujourd'hui l'attention de grand nombre de chercheurs, et dans divers domaines. En 2002, Hinds et

Kiesler y consacre l'ouvrage intitulé « *Distributed work* ». Dans cet ouvrage, la notion de travail distribuée est présentée à la fois comme un travail qui suppose l'éloignement géographique et temporel (King et Frost, 2002), mais également la répartition du travail entre des personnes travaillant ensembles durant une période allant de semaines jusqu'à plusieurs années (Moon et Sproull, 2002).

Un travail distribué est également un travail collectif qui réuni de nombreux participants autour d'une activité commune. Ces participants forment ce que Turner et al. (2006) appellent de collectif distribué. Ils interagissent, échangent des informations, créent des relations, et mettent en place collectivement des pratiques qu'ils partagent. Experts ou novices, concepteurs ou utilisateurs, ces participants viennent d'univers différents. Ils partagent un intérêt commun et contribuent collectivement au travail qui les rassemblent (Benkler, 2002 ; von Hippel, 2005).

La distribution du travail pause cependant des interrogations sur la façon dont l'activité est organisée et gérée (Hinds et Kiesler, 2002 ; Schmidt, 2000). Selon Turner et al. (2006), un travail distribué est absent de toute gestion centrale :

> « *In distributed collectives, there is no central control, no central locus for technical and interactional innovations, no central arbiter of ethics, meaning, or representation, and no central awareness of its implications*. » (Turner et al., 2006, p. 97)

Dans le développement de logiciels, Kiesler et Cummings (2002) montrent que la distribution du travail est gérée par deux mécanismes : la modularisation[6] et les procédures de contrôle de versions[7]. Ces mécanismes permettent aux participants de comprendre leurs rôles, de réduire les erreurs et de travailler de façons autonomes:

> « *Task decomposition and version control help people understand their goals and those of others, reduce error, and reduce the need to redo work.* » (Kiesler et Cummings, 2002, p. 69)

D'après Grinter, Herbsleb, et Perry (1999), la modularisation a un rôle important dans la gestion du travail distribué, elle n'est cependant pas suffisante. En effet, Grinter, Herbsleb, et Perry (1999) montrent que les projets distribuées présentent souvent des difficultés de

---

[6] Selon Parnas (1972), la modularisation désigne la décomposition des tâches. Dans le cadre du développement logiciel, la modularisation consiste à découper le code en plusieurs parties a fin de le traiter de façons indépendante.

[7] Les procédures de contrôles de versions assurent que les contributions soient conformes aux exigences et aux normes de la communauté Mozilla.

coordination, de confiances, de communications et d'interactions. Ces problèmes sont encore plus critiques dans le cas du développement de logiciels, lorsque le traitement du code en plusieurs parties engendre des problèmes d'incompatibilités lors de la phase d'intégration (Grinter et Herbsleb, 1999).

I.B.2. Distribution du travail et résolution de problème

La résolution de problème est une activité qui requiert un haut niveau d'expertise souvent menée de manière fortement distribuée. Davis et Smith (1983), la décrivent comme une activité coopérative, décentralisée, très riches en connaissances:

> « *Three defining characteristics of distributed problem solving are that it is a cooperative activity of a group of decentralized and loosely coupled Knowmedge-sources (Ks).* » (Davis et Smith, 1983, p. 67)

Les auteurs montrent que la distribution permet de résoudre les problèmes en le décomposant. D'après eux, lorsque les informations disponibles sont insuffisantes, la décomposition du problème permet la production collective de solutions grâce au partage de connaissances en provenance de sources différentes :

> « *The Knowmedge-sources (Ks) cooperate in the sens that no one of them has sufficient information to solve the entire problem, so a mutual sharing of information is necessary to allow the group as a whole to produce an answer.*» (Davis et Smith, 1983, p. 67)

Pour Lakhani (2006), la distribution du travail de la résolution de problèmes suppose, non seulement que les participants se trouvent dans des endroits géographiquement éloignée, mais également que des acteurs, n'ayant pas de connaissance initiale sur le problème soient impliqués dans la résolution (Lakhani, 2006). Distribuer un problème présente, dans ce sens l'avantage de pouvoir profiter d'un réseau d'acteurs diversifiés et de l''ouverture à de nouvelles connaissances.

Ainsi, et en se basant sur l'hypothèse de l'hétérogénéité des connaissances (Von Hippel, 1994), Lakhani (2006) développe le concept de «*Broadcast Search* », pour expliquer l'importance de la distribution du problème. Il montre que la distribution ou la diffusion du problème repose sur quarte éléments :

- *la définition de problème* doit tenir compte du fait que le problème soi exposé à un vaste espace de solution. Ceci exige que les phases de définition du problème et de l'identification de la solution soient séparées (Simon, 1973);
- *l'hétérogénéité des acteurs* est l'élément qui permet de résoudre efficacement un problème. En effet, pour augmenter la probabilité que le problème soit résolut, il doit être diffusé auprès d'un grand nombre d'acteurs pas nécessairement liés aux mêmes domaines scientifiques et techniques. Le concept de «*Brodcast Search* » repose sur la diversité des potentiels des acteurs externes. Pour Lakhani (2003), ce n'est pas le nombre de contribution qui permet de résoudre efficacement un problème, c'est plutôt la diversité des approches proposées pour la création de solutions nouvelles ;
- *le rôle des utilisateurs* est important dans la résolution de problème. Faire participer des utilisateurs permet à ces derniers de déclarer les problèmes qu'ils rencontrent et de participer au test des solutions conçues à cet effet. Selon Riggs et Von Hippel (1994), faire participer des utilisateurs permet d'identifier et de mieux cerner leurs besoins et par la suite assurer la réussite de l'activité innovatrice ;
- *Une structure de motivation* doit être mise en place pour attirer divers acteurs dans le but de créer des solutions aux problèmes rencontrés. Les motivations peuvent être intrinsèques ou extrinsèques. Ce critère sera développé au second chapitre.

Les notions de distribution et de résolution de problème sont étroitement liées. Lorsqu'elles sont associées elles renvoient à des aspects tels que l'éloignement géographique, l'hétérogénéité des connaissances, et la répartition du travail (modularisation). Toutefois, la distribution des problèmes révèle des difficultés quand à la capacité de coordonner l'activité. Ces difficultés semblent être dépassées dans des contextes de résolution de problèmes fortement distribués tels que le développement Open Source. D'après Gasser et Ripoche (2003) l'ouverture du code source joue un rôle central dans l'organisation de l'activité dans le cadre des projets Open Source. Elle donne la possibilité aux participants d'échanger et d'interagir continuellement (Gasser et Ripoche, 2003). Dans une étude sur le projet Open Source Mozilla, nous avons même identifié une variété de modes d'interactions (Dalle, Masmoudi et Den Besten, 2009).

Section 2. Développement Open Source et résolution de problème distribuée, le cas du projet Mozilla

La résolution de problème dans le cadre du développement Open Source, et particulièrement de Mozilla, représente une réelle situation de travail de résolution distribuée. En effet, la particularité du développement Open Source repose sur le fait que le travail de résolution soit effectué par plusieurs participants dispersés géographiquement, ayant des connaissances et de compétences diversifiées et utilisant Internet comme le seul moyen de communication.
A la différence des méthodes de développement logiciel traditionnel, le développement Open Source se démarque par des méthodes de fonctionnement spécifiques. Nous nous intéressons dans cette partie au domaine de l'Open Source, dont l'activité de développement constitue, comme nous allons le voir, un exemple d'une activité distribuée. Nous étudions, en particulier, l'activité de résolution de problème dans le cadre du projet Mozilla. Nous présenterons, ainsi dans une première partie, certains aspects qui caractérise le développement Open Source. Nous nous focalisons dans une seconde partie au cas de Mozilla.

II.A. Les caractéristiques du développement Open source

Le succès des projets tel que Mozilla, Firefox, Linux ou Apache atteste aujourd'hui de l'ampleur du phénomène Open Source et de son importance. Cet intérêt est lié au développement des nouvelles technologies de l'information et de la communication (désormais appelé NTIC) et particulièrement d'Internet au milieu des années 1990. Dans ce contexte, les nouvelles technologies présentent non seulement le principal moyen de communication, mais constitue également une réelle infrastructure d'organisation et de coordination. En effet, dans le cadre du développement Open Source, les participants aux projets interagissent volontairement et virtuellement sous forme de requêtes dans des forums. Ces requêtes prennent la forme de questions adressées aux autres membres dans l'objectif de rechercher des informations, des conseils, et d'expertises. Ils peuvent provenir des développeurs des logiciels mais aussi de ses utilisateurs, qui vont signaler les erreurs ou les imperfections des programmes existants. Nous présentons dans ce qui suit le développement Open Source ainsi que certains de ses aspects qui sont liée à la résolution du problème.

II.A.1. Définition

C'est en 1983, que Richard M. Stallman un chercheur en informatique au MIT (Institut de Technologie au Etat Unis) élabore un système informatique complet et libre, appelé « GNU[8] ». Il créé une fondation pour le logiciel libre « FSF[9] », rédige une licence adoptée à son système, la « GPL[10] » et défini un logiciel libre comme « *un logiciel qui fourni avec l'autorisation pour quiconque de l'utiliser, de le copier et de le distribuer, soit sous une forme conforme à l'originale, soit avec des modifications, et ce, gratuitement ou contre un certain montant* » (Stallman, 1983).

Selon cette définition, la FSF distingue quatre libertés qu'un logiciel libre possède et offre à ses utilisateurs/développeurs :

- *la liberté d'exécuter le programme, pour tous les usages ;*
- *la liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins ;*
- *la liberté de diffuser ;*
- *la liberté de modifier et de publier ces modifications.*

Un logiciel libre est la traduction de *« Free Software »* qui est également appelé logiciel *« OpenSource »*. Ces notions ont largement fait débat dans la littérature. Certains les considèrent comme des notions séparées (Free Software Foundation, 2005), d'autres comme synonymes (Von Hippel et Von Krogh, 2003). Nous adhérons au derniers avis nous utilisons ainsi les expressions logiciel Open source et logiciel libre de manière similaire.

---

[8] Le GNU est un projet fondé par Richard Stallman en 1984 dans le but de créer un système d'exploitation libre. GNU signifie en anglais « GNU's NOT UNIX ». Le projet GNU s'est développé avec un esprit coopératif et la libre diffusion de la connaissance. Conçu pour être compatible avec le système d'exploitation UNIX, très répondu à l'époque, GNU se définit comme une plate-forme dédiée à l'ensemble des logiciels libres.

[9] La Free Software Foundation FSF est une organisation à but non lucratif fondée par Richard Stallman. Sa fonction est d'assurer la promotion et la défense des logiciels libres au niveau international. Elle a donné naissance à la licence « GPL »

[10] La GPL est une licence, un document juridique qui définit le mode d'utilisation, d'usage et de diffusion de logiciels libres.

En tant qu'objet technique, Bastien (2001) défini un logiciel libre « *comme un ensemble d'automatismes programmés pour traiter des donnés* ». Selon Bastien (2001), le logiciel est lui même un ensemble de données se présentant sous deux formes :

- le « binaire » ou « l'exécutable » représente le logiciel tel que le système d'exploitation pourra l'interpréter ;
- le « code source » représente le logiciel tel qu'un être humain pourra l'interpréter, pourvu qu'il connaisse le langage de programmation dans lequel le logiciel a été écrit.

Foray et Zimmerman (2001) pour leur part, mettent en avant le caractère bénévole et coopératif du développement des logiciels libres. En étudiant le cas du système d'exploitation Linux, ils définissent un logiciel libre comme :

> « *Un logiciel dont le code source, c'est-à-dire la série d'instructions qui forme le programme avant la compilation, est rendu ouvertement disponible et ne peut faire l'objet d'une appropriation privative. Le développement des logiciels libres est fondé sur le volontariat et le bénévolat des participants, dans un mode d'organisation coopératif qui s'appuie largement sur les commodités organisationnelles issues d'Internet* ». (Foray et Zimmerman, 2001, p. 81)

De ces définitions émergent deux aspects qui caractérisent les logiciels libres et sur lesquels sont développement est fondé, d'abord la libre disponibilité des codes sources, ensuite la contribution de participants qui n'anticipent pas de rémunérations pour leurs efforts dans l'amélioration et la modification du logiciel (code source).

Pour Dalle et al. (2005) ce n'est pas seulement le caractère bénévole qui distingue le développement Open Source mais c'est surtout la distribution géographique de ses participants:

> « *It is the scale and speed of F/LOSS development work and the geographical dispersion of the participants, rather than the voluntary nature of the latters' contributions, properly should be deemed historically unprecedented.*» (Dalle et al., 2005, p. 397).

Dans la même veine, d'autres travaux attestent que les logiciels libres sont d'abord le fruit d'un travail coopératif entre de nombreux participants, qui ne se connaissent que de

manière virtuelle. Ces participants peuvent être développeurs mais également des utilisateurs.

Dans le cadre de la résolution de problèmes de logiciels (ou de défaut), Lakhani et Von Hippel (2003) expliquent que les participants collaborent car la solution recherchée par l'un est « sur étagère ». En effet, la solution recherchée peut être connue rapidement par un autre contributeur, qui n'aura donc pas à fournir un effort très grand pour la retrouver et l'envoyer.

Raymond (1999) montre également que l'ouverture du code source permet au logiciel d'être développé par des spécialistes mais au même temps par des utilisateurs qui ont des méthodes de travail très variées. L'ouverture et la diversité des participants font que les problèmes soient identifiés et résolus rapidement. Raymond (1999) souligne le rôle que jouent les utilisateurs dans le développement des logiciels libres. Pour Raymond, intégrer plus d'utilisateurs dans le travail de développement d'un logiciel permet d'identifier plus de problèmes, les corriger plus rapidement, et développer par la suite rapidement une version la plus adaptée. Leurs rôles sont encore plus pertinents, lorsque les utilisateurs sont des co-développeurs.

> « *More users find more bugs because adding more users adds more different ways of stressing the program. This effect is amplified when the users are co-developers. Each one approaches the task of bug characterization with a slightly different perceptual set and analytical toolkit, a different angle on the problem….In the specific context of debugging; the variation also tends to reduce duplication of effort.* » (Raymond, 1999, p4)

II.A.2. Le développement Open Source : un développement distribué pour la résolution de « bogues »

Dans la littérature (Crowston et al., 2006 ; Mockus et Herbsleb, 2002 ; Reis et de Mattos Fortes, 2002), la résolution des problèmes est l'une des caractéristiques fondamentale du développement Open Source. Elle représente d'une manière générale une part importante de tout projet de développement de logiciel. D'après Brooks (1995), 40% de l'activité de développement de logiciels est consacré à la résolution de problèmes ou de bogues.

En pratique la tâche de produire un logiciel est très complexe. Il est donc difficile de produire un code ne contenant aucun défaut. Selon Ripoche (2006), les problèmes que l'on rencontre lors du développement d'un logiciel sont non seulement liés à « l'infaillibilité » des développeurs, mais surtout au fait qu'un programme logiciel est un objet qui évolue

constamment et que cette évolution crée des interdépendances suites aux modifications ou à l'ajout d'une nouvelle fonctionnalité. (Ripoche, 2006).

Pour Crowston et Scozzi (2004), les logiciels Open Source sont principalement développés par des équipes distribuées. Les développeurs contribuent en communiquant exclusivement par Internet et utilisant les forums de discussion ou encore les messageries électroniques, comme supports des échanges.

La distribution est également la conséquence de l'ouverture du code source. Lakhani et Von Hippel (2003) montrent que la mise à dispositions du code source permet de diffuser le problème à un nombre important de développeurs spécialisés, mais surtout à des utilisateurs qui ont des connaissances et des compétences diversifiées. Ce dernier point a été développé par Mockus et Herbsleb (2002) qui expliquent, en étudiant les projets Apache et Mozilla, que l'ouverture du code source engendre des temps de résolution et des taux de défauts moins importants en comparaison avec le développement traditionnel.

Les auteurs montrent également que l'efficacité de l'Open Source est liée aux développements simultanés et rapides de plusieurs versions de codes ainsi qu'à la possibilité de les tester rapidement et de les améliorer, grâce à la participation des utilisateurs.

Nous pouvons dire que le développement Open Source est une activité distribuée de part sa capacité à être menée de manière fortement distribuée et à exploiter différents niveaux d'expertises. Cette activité est également dirigée par un processus de résolution de problèmes ou de bogues.

En effet, l'ouverture du code source accorde la possibilité aux participants de le modifier par l'élaboration de requêtes. Lorsque des requêtes sont proposées, elles sont discutées et débattues au sein de la communauté, révisées et enfin intégrées si les modifications sont acceptables. Ces requêtes sont présentées sous formes de rapports de bogues ou de problèmes. Toute modification apportée au code source correspond ainsi à une requête, un rapport de bogue distribuée à l'ensemble de la communauté.

La distribution pause, cependant, un nombre d'interrogations sur le processus qui permet de la gérer l'ensemble des requêtes ou les bogues signalés par les participants. Selon Gasser et Ripoche (2003) et Ripoche (2006), gérer un bogue est un processus complexe composé d'un ensemble d'activités consistant à *détecter, décrire, répertorier, organiser, hiérarchiser, assigner, analyser, résoudre, tester et vérifier ces bogues*.

Pour Crowston et Scozzi (2004), gérer un problème consiste à gérer les dépendances entre tâches, ressources et acteurs. Ils montrent que le processus qui permet de gérer un bogue

est composé de six tâches (cf. annexe I.1) : la *soumission de bogue* (des rapporteurs (*Submitters*) soumettent un problème. La soumission peut être liée à des problèmes d'incompatibilités, des symptômes, ou des informations sur le code) ; l'*assignation du bogue* (après la soumission, les responsables du projet ou les administrateurs assignent le problème à un responsable ou dans certains cas les développeurs s'auto-assignent les bogues), l'*analyse du bogue* (d'autres membres de la communauté rejoignent le rapporteur et le responsable du problème pour discuter de ses causes, annoncer des problèmes semblables qu'ils rencontrent), la *fixation du bogue* (les personnes impliqués dans la résolution du bogue proposent des solutions provisoires (patches) ou des éléments de solutions) ; la *vérification* d'une solution (patche) (la correction du bogue peut être suivie par le test et l'intégration du patche); et la *fermeture* du problème. Pour Crowston et Scozzi (2004), les tâches d'assignation et de vérification sont des mécanismes de coordination généralement utilisés pour gérer les dépendances entre les tâches et les acteurs[11]. Les auteurs expliquent que de tels mécanismes sont peu fréquents dans les projets Open Source. Ils émergent spontanément sur la base des compétences des experts, des descriptions et des analyses fournies par les autres contributeurs :

> « *The above analysis provides some interesting insights on the bug fixing process for FLOSS development.…Only few bugs are formally assigned. Such a coordination activity seems rather to spontaneously emerge. Based on bug description and analysis, those who have the competencies autonomously decide to fix the bug. That activity is facilitated by the supplied backtrace and analysis often undertaken by several contributors.* » (Crowston et Scozzi, 2004, p.8)

Dans la même veine, Sandusky et Gasser (2005) étudie le cas de la communauté Mozilla. Ils présentent la gestion de problème dans Mozilla comme un processus socio-technique complexe. Dans ce processus la négociation joue un rôle important. Elle est employée pour traiter l'incertitude et la complexité, pour coordonner l'activité de résolution de problème, et pour supporter le travail collectif. Le rapport de bogue constitute le principal support :

> « *Negotiation is a basic social in software problem management process frequently employed to address uncertainty and complexity, coordinate activity, and support distributed collective sensemaking, using the bug report as its locus.* » (Sandusky et Gasser, 2005, p. 190)

[11] Nous développons la notion de dépendances dans le chapitre III.

Les auteurs montrent que le processus de gestion de problèmes dans Mozilla est composé de quatre phases (cf. annexe I.2) :

- dans une première phase les problèmes sont *identifiés et rapportés* : un utilisateur ou développeur rencontre une anomalie erreur lors de l'utilisation ou le test d'un logiciel (étpae1). Cette anomalie est signalée par la création d'un rapport de bug (étape 2). Le rapport de bogue constitue ainsi le support de l'activité de résolution. Il représente la relation entre l'information et l'activité qui sont tout les deux liée à la gestion de la résolution de problèmes: «*Bug reports are central artifacts within almost all software development communities and are a nexus of information andactivity related to both software problem management.* » (Sandusky et Gasser, 2005, p. 188) ;
- la deuxième étape consiste à *trier les rapports de bogues* et à les *affecter* aux experts : les rapports sont classés par ordre de priorités. Ils sont par la suite affectés ou « assignés » à des experts. Cette étape (détermination du niveau de priorité et l'assignation du rapport) est évaluée par un groupe de développeurs qui ont un rôle primaire dans la gestion de la résolution du problème ;
- une fois le problème rapporté et trié les développeurs, impliqués dans le travail collectif de résolution, s'engagent dans les tâches d'*analyse* ; de *fixation*, de *test* et de *développement*: Les experts (*Assignee*) étudient le problème, déterminent sa cause, consultent d'autres experts, coordonnent le travail, modifient, testent le logiciel et intègrent leurs modifications pour résoudre enfin le problème. A cette troisième phase, la négociation est un processus social fondamental dans la gestion de la résolution de problèmes.
- La dernière phase du processus de gestion du problème consiste à *vérifier* et *clôturer* le problème : la vérification du bug est effectué par le responsable assurance qualité (*Quality assurance*)[12]. Si la vérification est validée, le statut du bogue sera marqué « vérifié » et enfin « fermé » par le gestionnaire du ficher central (*central manager file*) ou l'ingénieur du logiciel (*software engineer*).

La littérature identifie plusieurs processus qui permettent de résoudre un bogue. Ces processus peuvent varier d'un projet à un autre. Dans l'étude de Mockus et Herbsleb (2002) sur les projets Apache et Mozilla, les auteurs observent que dans le projet Apache

[12] Le maitre de la plaforme (the platform master), responsable de la construction du logiciel ou l'équipe d'intégration.

chaque membre peut déclarer des bogues, proposer les contributions qu'ils souhaitent apporter, et que le processus d'intégration de ces contributions n'exige pas le respect de certaines normes ou qu'elles soient tout le temps vérifié. Dans le projet Mozilla, l'intégration des contributions est beaucoup plus strictes et nécessite qu'elles soient validées par les responsables du projet (cf. partie II.B.2.b).

Par ailleurs, le processus d'intégration des contributions ou plus globalement le processus de résolution de bogues n'est pas figé et varie en fonction des situations et de la nature même des bogues. Par exemple, dans le projet Mozilla, les contributions qui concernent des bogues de sécurité sont soumis à des super-revues (super review), tandis que d'autres ne le sont pas. Certains bogues sont affectés par les responsables des composants à des personnes qui se chargent de suivre le processus de résolution des bogues, d'autres sont gérées par des personnes qui s'auto affectent cette fonction sans qu'ils soient désignés par un responsable.

Nous allons d'ailleurs examiner le projet Mozilla, comme un cas d'étude concret qui va mettre en lumière les éléments théoriques que nous venons d'avancer. Cette investigation va permettre d'avoir une première vision sur la manière avec laquelle l'activité de résolution de bogues est organisée au sein de Mozilla.

II.B. Le projet Mozilla

Comme nous pouvons le constater, le projet Mozilla a fait l'objet de grand nombre d'études dans le domaine de l'Open Source et particulièrement dans le domaine de la résolution de problème distribuée (Sandusky et Gasser, 2005, Ripoche 2006, Mockus et al. 2002, Friedman, 2002). Ces études présentent Mozilla comme un projet de développement complètement distribué, dirigé par la résolution de bogues, ce qui nous intéresse particulièrement dans la mesure où nous souhaitons étudier une activité de résolution de problèmes distribués. Nous utilisons d'ailleurs le cas du projet Mozilla dans les chapitres suivants pour illustrer des aspects particuliers de notre approche.

II.B.1. Historique

Mozilla est crée en 1998 suite à la décision de Netscape de mettre à disposition librement le code source de son navigateur. Rapidement, des développeurs et utilisateurs à travers le monde utilisent et proposent des améliorations pour les différents produits. Ces développeurs et utilisateurs, qui ont été jusqu'à 2005 entièrement bénévoles, forment la

communauté Mozilla. Ils collaborent exclusivement sur Internet et par l'intermédiaire des outils de développement accessibles depuis un navigateur web.

Depuis sa création Mozilla a beaucoup évolué. Nous présentons cette évolution en cinq étapes :

- *La naissance de Mozilla* : En mars 1998, l'entreprise Netscape (*Netscape Communications Corporation*)[13] décide de dévoiler le code source de la version 4.0 de son navigateur nommé Communicator[14] et créer par la suite Mozilla. Cette décision venait suite à une chute importante des parts d'usage (de 80% en 1996 à 55% en 1997) de Communicator face à Internet Explorer, navigateur crée par Microsoft, qui à l'inverse connaissait une augmentation considérable des parts d'usage (de 5% en 1996 à 36% en 1997). Pour gérer le développement de Mozilla, Netscape a créée une structure nommée *Mozilla Organisation*. Cette structure est informelle et composée d'employés qui avaient pour rôle de coordonner l'activité de Mozilla en prenant en charge le maintien d'un module. En novembre 1998, Netscape est racheté par (*America Online*) AOL[15] pour la valeur de 4,3 milliards de dollars.
- *La création de la Mozilla Foundation* : En juillet 2003, AOL crée, un organisme à but non lucratif qui avait pour fonction de récupérer toutes les missions de Mozilla Organisation qui avait. Mozilla Fondation avait aussi pour mission de promouvoir le développement, d'assurer la publicité des produits Mozilla, et de préserver l'innovation sur Internet (Mozilla.org, 2003)[16].

---

[13] Netscape est une entreprise d'informatique américaine crée en 1994. Elle développe le premier navigateur Internet Mosaic au centre de recherche NCSA (National Center for Supercomputing Applications). (Wikipédia 2010)

[14] Communicator est la version 4.0 du Navigator. Navigator était un navigateur Web édité par la société Netscape. La première version de « Navigator » était lancée en octobre 1994 et commercialisé en décembre 1999. Il a dominé le marché au milieu des années 1990 et n'as pas résisté face au concurrent Internet Explorer. (Wikipédia 2010)

[15] AOL est une société américaine de services internet, ancienne filiale du groupe diversifié de médias Time Werner.

[16] « Mountain View, CA, July 15, 2003 – Mozilla.org, the organization that coordinates Mozilla open source development, today announced the launch of a new foundation that will continue to promote the development, distribution and adoption of the award-winning Mozilla standards-based web applications and core technologies, including the Gecko browser layout engine. The Mozilla Foundation will continue and expand on the efforts of mozilla.org, the group managing the daily operations of the

- *La création de Mozilla Corporation* : En août 2005, la Fondation Mozilla décide de créer une filiale appelée *Mozilla Corporation*. La Mozilla corporation s'occupe du financement et du développement de Mozilla. Elle se charge, plus spécifiquement de la gestion des partenariats technologiques et financiers avec des entreprises. Ces partenariats lui permettaient de rémunérer certains développeurs non bénévoles qui contribuaient au développement de Mozilla. Ajouter la partie sur les aspets prioritaires.
- *La création de Mozilla Messaging* : En février 2008, la fondation *fondation Mozilla* crée *Mozilla Messaging*. Cette nouvelle filiale a été dédiée à la promotion des solutions de messagerie Mozilla. A travers son site mozillamessaging.com, *Mozilla Messaging* consacre également une section au recrutement de nouveaux contributeurs en leurs proposant de participer à la promotion de leurs solutions messageries et particulièrement de Thunderbird[17]. En effet, la création de la filiale *Mozilla Messaging* a été souhaitée pour promouvoir le développement de Thunderbird qui a été délaissé de la part des membres de la communauté en faveur de Firefox.

II.B.2. Communautés Mozilla: Rôles et responsabilités

La communauté Mozilla regroupe l'ensemble des personnes qui contribuent au projet que ce soit en déclarant des problèmes, en réalisant des tests, en soumettant des propositions pour modifier le code source, où même en donnant de simples impressions sur le fonctionnement d'un logiciel. Ils contribuent au développement du projet via Internet et à travers de plusieurs outils de communication tels que les forums de discussion, les listes de diffusions et les messageries instantanées.

La communauté Mozilla est souvent considérée comme une communauté hybride, rassemblant à la fois, les avantages du développement commercial et Open Source[18]. Comme pour le cas de tous projets Open Source, les participants peuvent adhérer librement

Mozilla project since its inception…"It has been a long-standing objective of the Mozilla team to create an independent organization so we can continue to lead and innovate," said Mitchell Baker, Chief Lizard Wrangler at mozilla.org" » (Mozilla.org, 2003)

[17] Thunderbird est un client de messagerie électronique distribuée gratuitement par la fondation Mozilla.

[18] Nous utilisant ici le terme Open Source pour faire références aux projet typiques qui, contrairement à Mozilla, possèdent une structure de gouvernance informelle, ce qui veux dire que chaque participants à al droit de modifier le code source sans que cette modifier soie approuver par le responsable du module.

au projet Mozilla, sous simple inscription sur le site web du projet. Ils peuvent contribuer volontairement de manière individuelle, ou contre rémunération lorsqu'ils sont engagés par une entreprise pour intégrer le projet.

Au même temps, le projet Mozilla applique des systèmes de contrôles similaires aux systèmes utilisés dans le cadre des projets commerciaux (Mockus et al., 2002). En d'autres termes et comme Mike Beltzner (2007) le directeur du développement de Firefox l'exprime, Mozilla donne la liberté aux contributeurs de proposer un changement, de s'exprimer sur une proposition de changement, de soumettre une modification au code. Cependant, la décision de changer le code n'est pas accordée à tout le monde :

> « *In Mozilla Anyone can propose a change. Anyone can comment on a proposal for change. Anyone can submit a change to the code. Not everyone can approve a change.* » (Mike Beltzner, 2007)

En effet, la décision de modifier le code source, d'accepter des propositions de modifications, ou toutes autres décisions clés du projet sont accordées à l'organisation Mozilla (le groupe mozilla.org). La fonction du groupe Mozilla.org est équivalente à la fonction des dirigeants dans les projets commerciaux. Elle regroupe des développeurs ayant les compétences nécessaires pour structurer et organiser les activités des communautés. L'organisation mozilla.org (2008)[19] distingue plusieurs rôles:

- Le *staff* du projet (Mozilla.org Staff) regroupe six membres[20] qui ont pour rôle de superviser le développement du projet et lui fournissent une orientation générale. Ils supervisent également l'ensemble des outils utilisés par les contributeurs comme par exemple Bugzilla et CVS, l'organisation de la communauté, l'assistance des nouveaux développeurs, la création de règles générales et de procédures pour le projet. La majorité de ces membres travaillent au sein d'entreprises privées qui les rémunèrent pour organiser et maintenir l'infrastructure du développement de Mozilla. Il n y a pas de comité formelle au sein du Staff. Le travail est réparti de manière à ce que les membres supervisent les parties du projet dans lesquels ils ont plus d'intérêts et d'expertises. Néanmoins, tous les membres du staff sont impliqués dans les décisions qui concernent la résolution de bogues supportés par Bugzilla ;

[19] http://www-archive.mozilla.org/about/staff.

[20] Nous présentons en annexe I.1 la liste actuelle des membres du staff du projet Mozilla et de leurs associées (*Mozilla.staff.Associates*). Ces derniers interviennent pour soutenir les membres de staff en fournissant des informations pour les aider à prendre des décisions qui concernent des aspects spécifiques du projet.

- Les *Divers* sont les membres[21] qui gèrent quotidiennement le développement des différentes versions des produits Mozilla. Ils ont des connaissances précises sur l'ensemble des produits développés par le projet et leurs interdépendances. Leur objectif est de gérer le développement du projet en coordonnant les ressources disponibles et en indiquant les priorités du projet ainsi que les éléments qui serons et ne serons pas intégrés dans le produit final. Les Drivers interviennent également dans la résolution des bogues (prioritaires) en fournissant des conseils et des informations aux développeurs : « *The drivers provide guidance to developers as to which bug fixes are important for a given mozilla.org release* » (mozilla.org, 2008);
- Le *Module Owner* [22] ou responsable de module a pour fonction de contrôler le développement d'un module[23] particulier (désignée par le Staff). Le responsable de module s'occupe principalement de l'évaluation (révision) de la conception et de l'implémentation des patches proposés par les membres de la communauté[24]. Les Modules Owners veillent à ce que les contributions (les patches) soient conformes aux exigences et aux normes de la communauté Mozilla. Le changement du code n'est donc pas accordé à tous les contributeurs. Il est possible (accepté et validé) à condition que les règles de la communautés soient respectées et que les membres de la communauté soient convaincus des avantages de ce changement : «*Mozilla community has decided that it can't accept just any change to be integrated into the public central Mozilla code base. If you want your code to become a part of it, you need to follow rules. These rules are not like law, but basically you must convince people that your change is good.* » (Mozilla.org, 2009)[25];
- Les *Super-Reviwers* sont des *hackers*, des développeurs qui ont beaucoup d'expériences. Désignés par le *Staff* mozilla.org, les *Super-Reviwers* ont une vision globale du code du projet. Ils interviennent dans le but d'évaluer les conséquences

[21] Nous présentons en annexe I.2 la liste actuelle des Drivers dans mozilla.org (2011).

[22] La liste actuelle des modules et leurs responsables est présentée sur l'adresse suivante : http://www-archive.mozilla.org/owners.html

[23] Un module est un sensemble de fichiers sources qui forment un paquet ou un groupe coherent : « A module is a set of files that implement a piece of functionality that has reasonably defined boundaries. » (mozilla.org, 2008)

[24] Un patch est un élément de solutions proposées pour résoudre les bogues rencontrées impliquant la modification du code source.

[25] Https://developer.mozilla.org/en/Mozilla_Hacker's_Getting_Started_Guide.

de l’ensemble des modifications proposées sur l’état général du code. L’évaluation des *Super-Reviwers* suit généralement celle des *Modules Owners*, et concerne un type particulier de bogues (par exemple les bogues de sécurités.

- Les *Bugzilla Component Owners* ou responsables de composant Bugzilla[26] ont pour fonction de gérer la résolution de bogues déclarés et associés à ces composants. Ils s’occupent notamment de l’affectation des bogues aux responsables de la résolution (correct owners) qu’ils désignent eux même[27], d’assurer dans certains cas le test des contributions proposées (patches), et du suivi du processus de la résolution des bogues correspondent à chaque composants.

Cet aperçu des rôles au sein du projet Mozilla met en évidence le fait que le processus de développement est très organisé, contrairement à certaines idées en rapport avec le développement Open Source. Néanmoins, les décisions ne sont pas prises par un groupe restreint, comme pour le cas du développement commercial. La modularité du code source fait que les tâches, les responsabilités, et les décisions soient réparties à un nombre plus large de responsables à la tête de chaque module. Ces aspects sont particulièrement adaptés à un processus de développement distribué.

II.B.3. Une Réussite : Firefox Mozilla

Aujourd’hui Mozilla est considéré comme l’un des projets les plus importants avec une communauté de développeurs utilisateurs composés de plusieurs dizaines de milliers de participants, et plus de 100 sites (cf. figure 1.2).

La réussite de Mozilla se traduit à travers le succès de son produit phare Firefox depuis son introduction en 2005, jusqu'à aujourd’hui. En effet, des statistiques récentes[28] montrent que Firefox et Internet explorer sont les navigateurs les plus utilisés en 2011 (cf. figure 1.3).

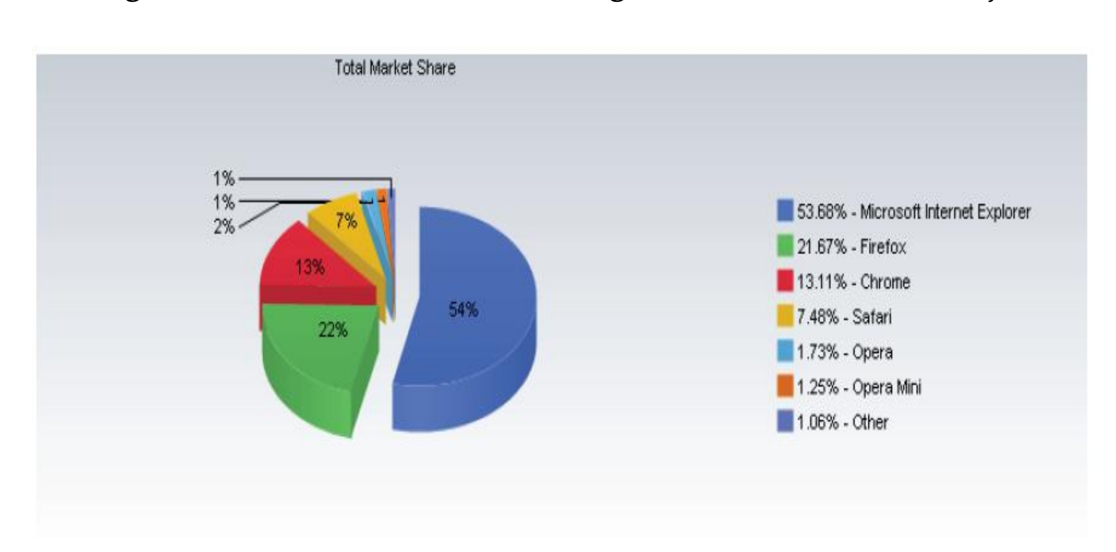
---

[26] La liste des composants et des responsables associés est présenté en annexe I.3 (A AJOUTER EN ANNEXE https://bugzilla.mozilla.org/describecomponents.cgi?product=mozilla.org)

[27] Comme nous l’avons avancé, certains contributeurs ou correct Owners s’auto- affectent des bogues sans qu’ils soient désignés par les component Owners ou autre membres de l’organisation mozilla.org. Nous détaillerons plus loin cet aspect de l’organisation de l’activité au sein de Mozilla.

[28] Selon market share le 18 juin 2011 (http://marketshare.hitslink.com/browser-market-share.aspx?qprid =1).

Figure 1.2 : Part de marché des navigateurs dans le monde en juin 2011[29]



Depuis son lancement, la part de marché de Firefox n'as pas arrêté de grimper, passant de 15,45% en juin 2007 à 23, 81% en juin 2010 (cf. figure 1.2), avec un nombre de téléchargement de la version Firefox 4 dépassant les 5 millions 24 heures après la date de son lancement, le Mars 2011. C'est sur les marchés Européens et français que Firefox a eu le plus de succès sa part sur ces deux marchés égalisant celle d'Internet Explorer en juin 2011 (cf. tableau 1.1).

[29] Source http://marketshare.hitslink.com/browser-market-share.aspx?qprid =1.



Figure 1.3 : Evolution de la part de marché de Firefox dans le monde[30]

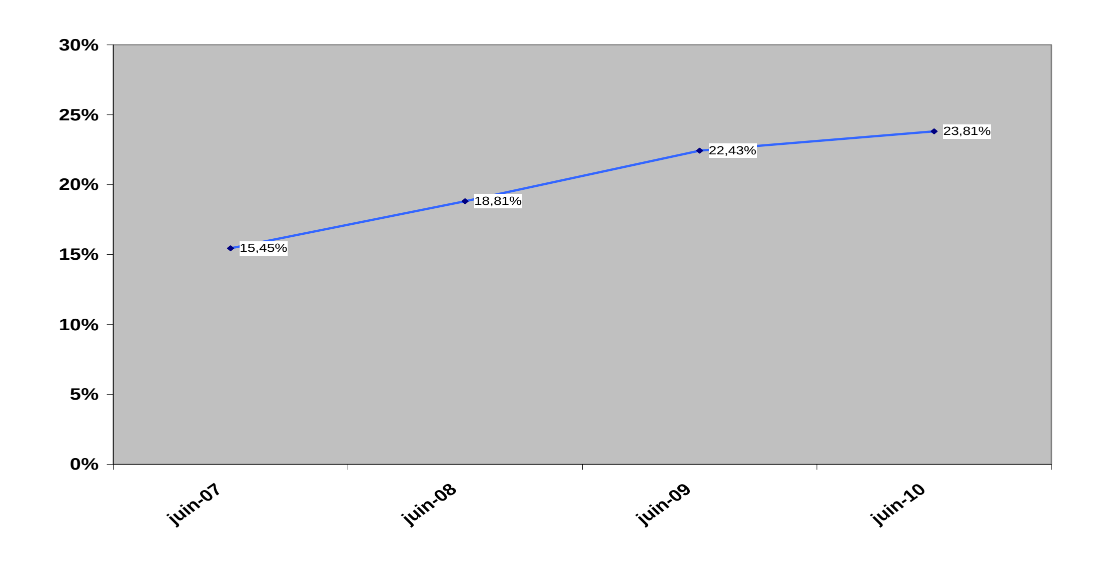


Tableau 1.1 : Les parts de marché des navigateurs en Europe, en France et dans le monde (Juin 2011)

	Internet Explorer (Microsoft)	Firefox (Mozilla)	Chrome (Google)	Safari (Apple)	Opera (Opera Software)	Autres
Europe[31]	35,47 %	34,67 %	19,96 %	5,3 %	3,91 %	0,68 %
France[32]	36,54 %	35,63 %	18,83 %	7,29 %	1,09 %	0,62 %
Monde[33]	53,68 %	21,67 %	13,11 %	7,48 %	2,98 %	1,06 %

30 Source : http://marketshare.hitslink.com.

31 Source : http://gs.statcounter.com.

32 Source : http://gs.statcounter.com.

33 Source : http://marketshare.hitslink.com.

II.B.4. Bugzilla : un outil incontournable

Pour créer un espace favorable à la distribution et au travail collaboratif, la communauté Mozilla développe un ensemble d'outils de communications accessibles sur Internet. Parmi ces outils, les plus utilisés sont le logiciel de gestion de versions CVS[34] (*ConcurrentVersions System*), le système de support à la discussion synchrone l'IRC[35] (*Internet Relay chat*) et le système de suivi de bogues *Bugzilla*.

Nous nous intéressons particulièrement à l'outil *Bugzilla*, puisque il s'agit d'un système qui permet aux participants de suivre le processus de résolution de bogues. Il est également utilisé pour l'attribution des tâches aux développeurs, mais aussi pour assister la coordination du travail. De plus *Bugzilla* sert de support de communication, il regroupe l'ensemble des bogues qui se rapporte à Mozilla sous forme de rapport issu de la communauté.

Bugzilla occupe une place essentielle dans le projet et fait partie intégrante du processus de son développement puisque les décisions en rapport avec le traitement des bogues sont prises de manière collaborative et distribuée via cet outil. De janvier à juillet 2011, 16826 bogues sont marqués résolus dans *Bugzilla*. De plus, *Bugzilla* est utilisé par 985 autres organisations privées et projets Open Source. Nous citons, par exemple, dans l'Open Source, les projets Linux Kernel[36], Gnome[37], Apache[38], Eclipse[39], et dans le privé, les entreprise Nokia, The New York Times, Yahoo, etc.

---

[34] Le CVS est un dépôt de données qui enregistre toutes les modifications apportées au code, les noms des auteurs, les dates de modifications, les documents utilisés, etc. Il permet aux développeurs et de gérer les différentes versions des logiciels développées dans le cadre du projet en s'assurant que les changements apportés ne provoquent pas des défauts au niveau du fonctionnement général de chaque logiciel.

[35] Le système de support de discussion synchrone IRC est un outil qui permet d'organiser de manière synchrone les discussions entre les participants via différents canaux de communication. Ces discussions portent généralement sur les décisions qui concernent

[36] http://bugzilla.kernel.org/

[37] http://bugzilla.gnome.org/

[38] http://issues.apache.org/bugzilla/

[39] http://bugs.eclipse.org/bugs/

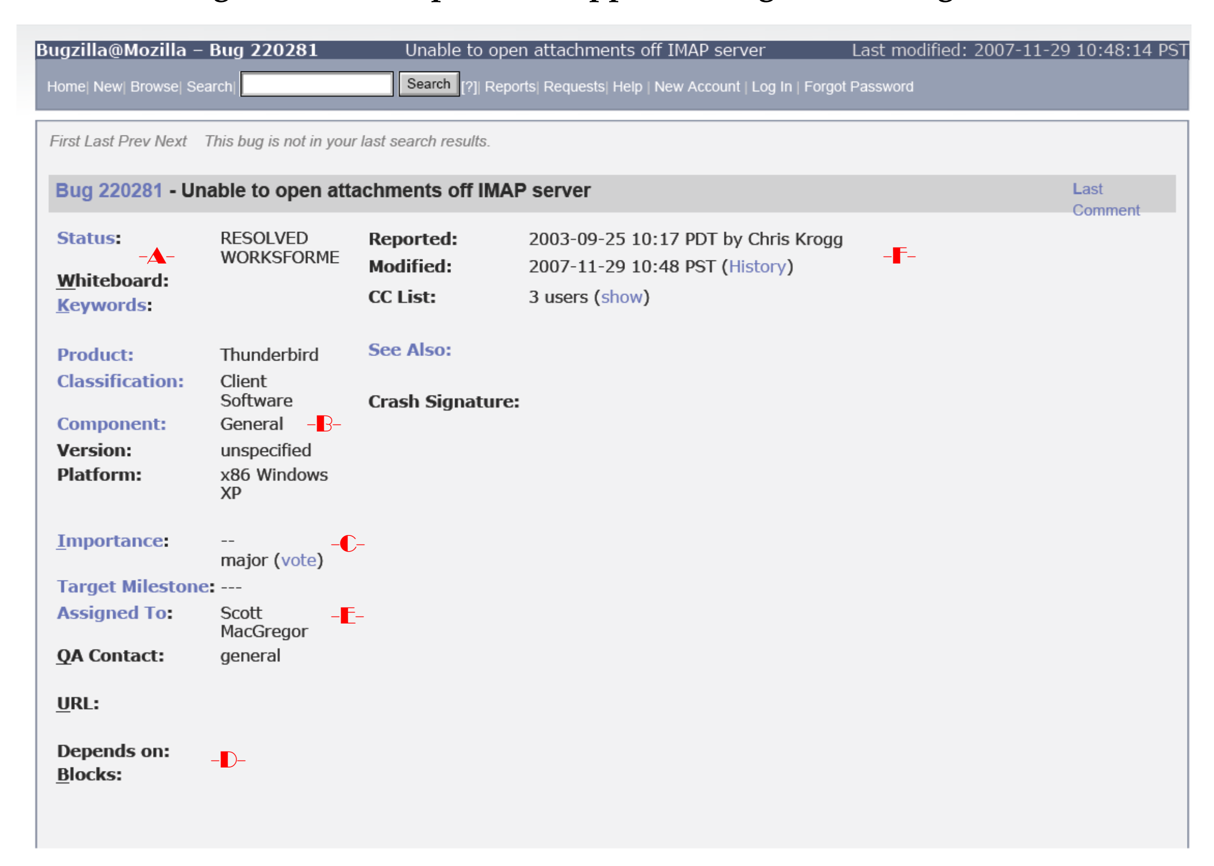
II.B.4.a. Descriptif du rapport de bogue dans Bugzilla

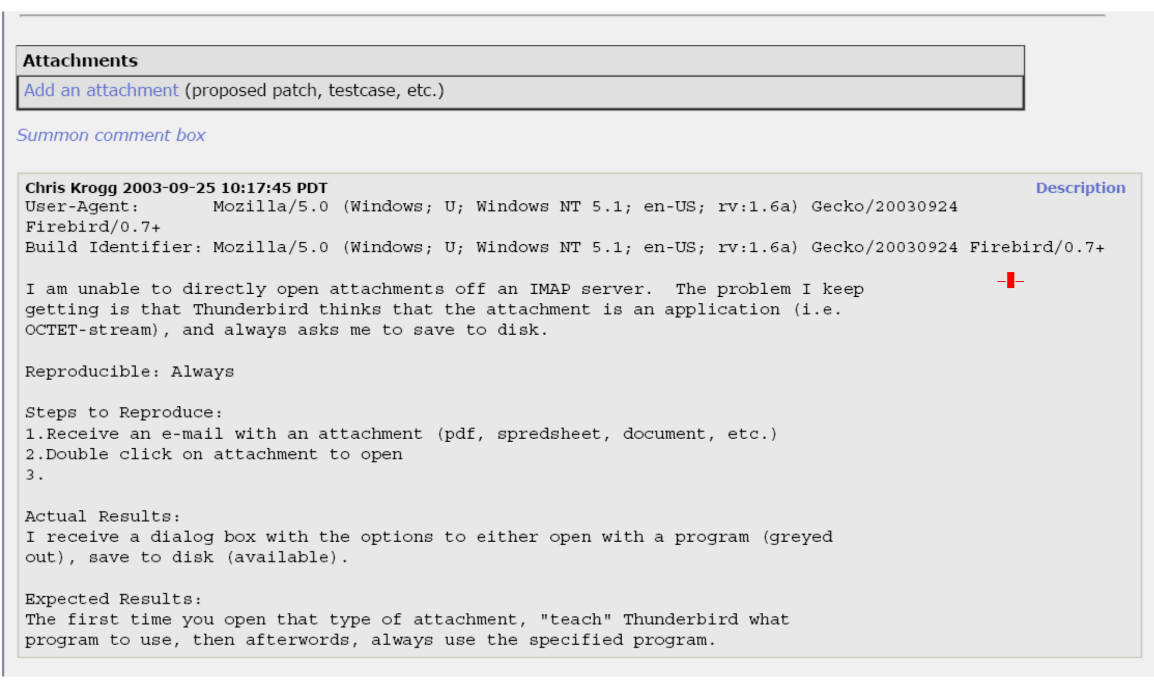
Les rapports de bogues sont accessibles librement via l'adresse web www.bugzilla.mozilla.com. Les participants peuvent donc déclarer des bogues, consulter les bogues en cours, et contribuer même à la résolution des bogues par la simple création d'un compte sur Bugzilla.

Dans *Bugzilla*, chaque bogue est présenté dans un rapport (une page web), identifié par un numéro. Un rapport de bogue regroupe, en deux parties, les différentes informations relatives au bogue (cf. figure 1.4) :

- La première partie présente les informations qui retracent les changements apportés au bogue, telles que, le statut actuel du bogue (–**A**–:*status, resolution*), le composant concerné (–**B**–: *componet, module*), l'importance du bogue (–**C**–: *Priority, severity*), le niveau de dépendance du problème avec d'autres modules (–**D**–: *depends_on*), l'identité actuelle de la personne responsable de suivre le bogue (–**E**–: *bug assignee*), le nom de la personne qui a déclaré le problème (–**F**–: *bug reporter*), ainsi que la liste des documents attachés au bogue (patches, captures d'écran, etc.).
- La deuxième partie présente une liste de messages commentaires échangés entre les développeurs liées au bogue. Ces commentaires, sous forme de messages textuels (–**I**–), sont organisés par ordre chronologique indiquant l'identité de l'émetteur. Les contenus des messages sont variables. Certains messages comportent une description du problème, d'autres présentent des instructions pour exécuter un programme, tester un patch, ou signalent un bug comme identique à un autre bogue (*bug duplicates*). Les messages varient aussi en nombre, ceci dépend de plusieurs facteurs tels que la complexité du problème, l'intérêt et l'importance qui leur est accordés.

Figure 1.4 : Exemple d'un rapport de bogue dans Bugzilla





II.B.4.b. La résolution de bogue dans Bugzilla : un processus formalisé

Dans *Bugzilla* chaque développeur est libre de déclarer des bogues à corriger ou de proposer de nouvelles fonctionnaliser pour résoudre d'autres bogues. Lorsque ces développeurs décident de travailler sur un rapport de bogue particulier, ils doivent le signaler sur la page web du bogue en rajoutons un commentaire. C'est à travers l'échange de commentaires que les solutions sont proposées, discutées, validées et intégrées. Ces échanges ne sont, cependant, pas totalement libre et subissent une sorte de formalisation.

La figure 1.5 donne une description de la manière dont l'activité de résolution de problème est organisée dans *Bugzilla*. Ainsi, la résolution de problème est composée de plusieurs étapes :

1/ *Analyse et tri* : lors de la première étape un rapport est crée par la déclaration d'un bogue dans Bugzilla. Le bug va ainsi avoir le statut de « NEW » ou de « UNCONFERMED ». Cette distinction dépend de la position (rôle) qu'occupe la personne qui a déclaré le problème. Si le rapporteur est reconnu par les autres développeurs, le bug est identifié avec le statut NEW. Lorsque le rapporteur se situe à la périphérie le bug occupe le statut UNCONFIRMED.

2/ *Attribution* : la deuxième étape consiste à attribuer le bogue à un développeur. L'attribution du bogue est la fonction du responsable de la composant concerné (*Bugzilla Component Owners*). Ce dernier choisi le développeur qu'il juge compétent, et lui affecte le bogue. Si le développeur accepte cette offre, il le signale dans le rapport en remplissant le champ (Assigned to), le statut du bug passe ainsi de UNCONFIRMED/NEW à ASSIGNED.

3/ *Résolution* : Dans cette troisième étape, les développeurs proposent individuellement ou collectivement des éléments de solutions sous forme de patches attachés au rapport de bogue. Ces patchs sont testés, discutés, et débattues par les autres participants. A la suite de cette étape le statut du problème passe à (RESOLVED). Ceci n'implique pas dans tout les cas la résolution technique du problème (RESOLVED FIXED).

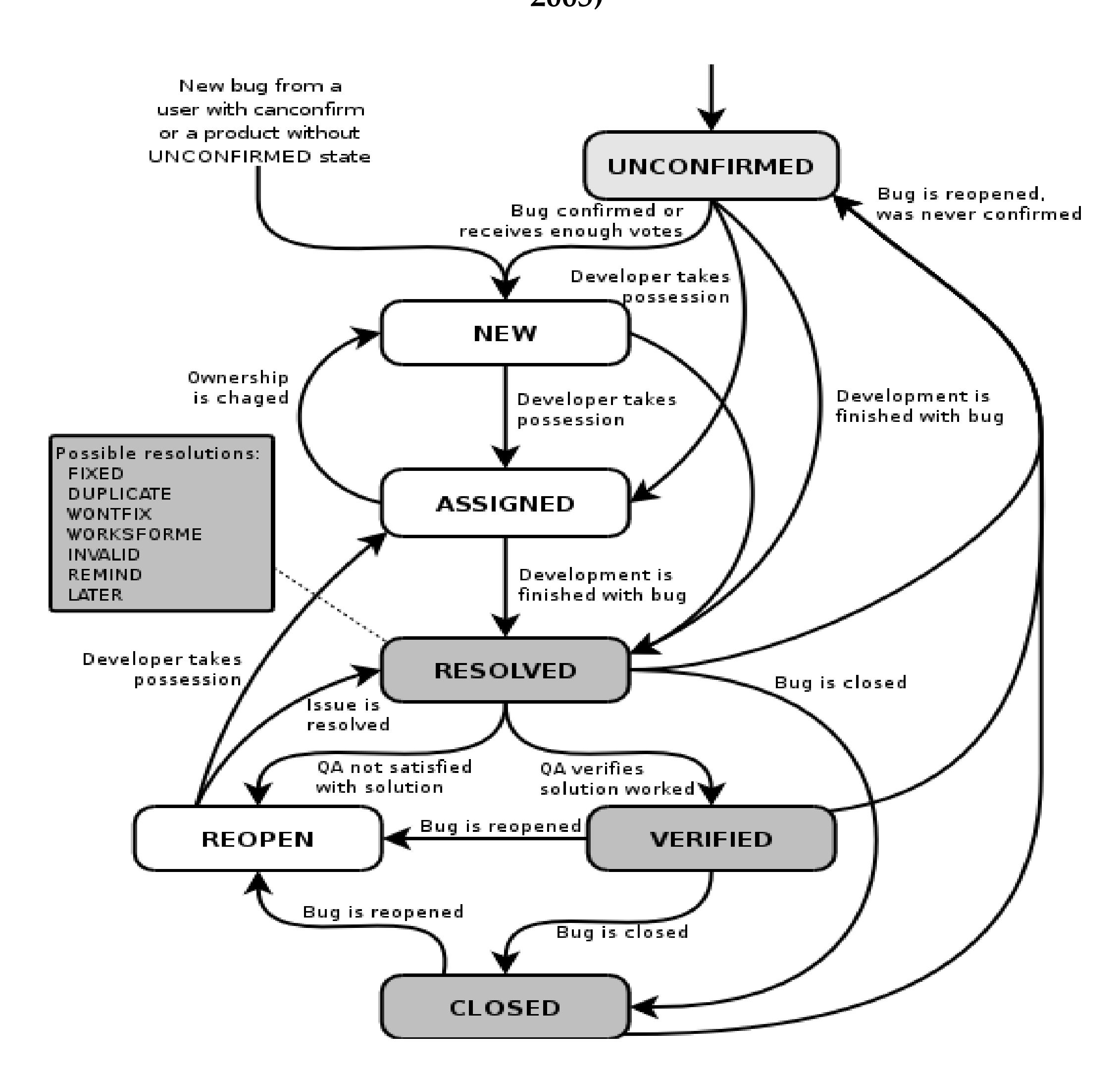
4/ *vérification* : Le bug est marqué (VERIFIED) lorsque la solution proposé est évaluée au travers une procédure stricte de vérification afin de s'assurer que cette solution (patche) permet de corriger le bogue sans affecter la qualité du code du projet. Dans le cadre de Bugzilla, la vérification du code est la fonction du (*reviwer*) qui est généralement

le responsable du module (*module owner).* Ce dernier examine la contribution soumise (patche) et l'intègre dans le code si elle est acceptée. Pour certains modules, avant d'être intégrées définitivement, les contributions doivent aussi être évaluée par les (*super-reviewers*).

5/ *Fermeture* : Après vérification, si aucune modification ne doit être apportéé, le statut du bug passe à CLOSED.

6/ *Réouverture* : Dans certain cas, le problème ré-apparaît. Plusieurs raisons peuvent l'expliquer comme l'inefficacité de la modification proposée ou la dépendance avec d'autres bogues.

Figure 1.5 : Processus de résolution de problème dans Bugzilla (Bugzilla Team, 2005)



Section 3. Synthèse et questionnement

Les travaux sur la résolution de problèmes ont souvent été abordés dans la littérature. Ils ont pour origine les recherches en sciences cognitives et en intelligence artificielle (Newell et Simon, 1972/1979). Pour Newell et Simon (1973) résoudre un problème consiste à chercher une solution dans un espace de problèmes. Cette approche se base sur les connaissances existantes dans cet espace ainsi que sur des procédures prédéfinis.

Néanmoins, Les récents travaux sur ces questions (Lakhani, 2006 ; Moon et Sproull, 2002) expliquent que la complexité des problèmes impose l'élargissement de l'espace de résolution vers de nouvelles compétences, d'ou l'approche de la distribution de problèmes (Crowston et al., 2006 ; Mockus et Herbsleb, 2002). Cette approche consiste à élargir l'espace de la résolution par le biais de la distribution de la résolution.

La présentation du développement Open Source en tant que modèle distribuée nous a permis de retenir qu'il repose principalement sur l'ouverture du code. Cette ouverture, permet à un plus grand nombre d'agents autonomes, dispersées géographiquement de proposer des solutions.

D'un autre coté, plusieurs études ont constaté les effets limitant de la distribution, en particulier dans le cadre d'une activité complexe qui requiert un niveau de coordination élevé (Crowston, 1997 ; Olson, 2000). Dès lors, le développement Open Source semble éviter de manières surprenantes ces difficultés.

Une question assez fondamentale consiste alors à comprendre comment les problèmes sont résolus dans le cadre d'un développement décentralisé qui fait participer un grand nombre d'agents autonomes. Ceci nous amène à nous interroger ***sur l'organisation du développement des logiciels libres, du point de vue de leurs structures et des mécanismes de coordination qu'elles mettent en place pour la résolution de problèmes.*** Ces interrogations feront l'objet des prochains chapitres.

CHAPITRE II- STRUCTURATION ET ORGANISATION DE L'OPEN SOURCE

TABLE DES MATIERES

CHAPITRE II- STRUCTURATION ET ORGANISATION DE L'OPEN SOURCE

Dans un second chapitre de ce travail, notre objectif, comme nous venons de l'expliciter dans la dernière section du premier chapitre, est de s'interroger sur l'organisation conduisant à la résolution de problème dans le cadre des communautés Open Source.
Etant dotées de caractéristiques tels que l'ouverture, la distribution du développement et la complexité du produit résultant, se pose la question de savoir *quel mode organisationnel caractérise les communautés Open Source. Les communautés Open Source sont-elles organisées comme les autres formes traditionnelles ou ont-elles un mode qui lui sont propres ?*
Porter notre attention sur l'organisation du développement Open Source, nous conduit ainsi à s'intéresser, dans une première section de ce chapitre, aux communautés et plus spécifiquement à la littérature qui étudie les communautés Open Source. Nous nous attachons dans une seconde section à présenter les travaux qui traitent de la structure de ces communautés. La dernière section, comme pour le premier chapitre, nous concluons par une synthèse des apports théoriques développés dans ce chapitre et les questionnements qu'ils suscitent.

Section 1. Les communautés Open Source : Apports et limites de la littérature sur leurs caractérisations

La littérature souligne l'existence d'une grande variété de formes organisationnelles développant des logiciels libres (Shah, 2006) et propose même une typologie permettant de prendre en considération cette variété (Benkeltoum, 2009; Henkel, 2004). Malgré cette variété, la notion la plus utilisé pour désigner ces formes est *la communauté* (O' Mahony, 2007 ; Henkel, 2003).

Après une recension des différentes définitions de la communauté (I.A) et de sa typologie (I.B), nous exposerons dans une seconde partie de cette section (I.C) les

différentes caractéristiques attribuées au concept de la communauté Open Source et notre positionnement.

I.A. Définition des communautés

La notion de communauté a été considérée différemment selon les disciplines. Ceci fait émerger diverses définitions qui mettent en avant des caractéristiques distinctes, et contribuent à l'élaboration d'une typologie des communautés.

En sociologie, Hillery (1955) défini une communauté comme « un groupe de personnes qui partagent des interactions sociales, des liens, et un espace commun ».

Karp et al. (1977) proposent une définition de la communauté en mettant en exergue trois caractéristiques : le maintien d'une interaction sociale, des valeurs partagées et un espace géographique délimité.

Ces définitions classiques de la communauté sont intéressantes elles ne présentent pas, cependant, la nature des relations qui peuvent exister entre les acteurs qui la compose. Selon Brint (2001) ces relations se caractérisent par la fidélité, l'affection et des valeurs partagées. Il propose ainsi la définition suivante de la communauté:

> « *Aggregates of people who share common activities and/or beliefs and who are bound together principally by relations of affect, loyalty, common values, and/or personal concern.*[40]» (Brint, 2001, p8).

Selon (Muniz et O'Guinn, 2000), une communauté se distingue par ses normes sociales (valeurs, pratiques, codes, règles, rites, etc). A partir de ces normes, les membres de la communauté développent une identité et des valeurs communes. Selon Paul Muller (2004), ces éléments facilitent l'action collective. Elles décrivent de manière générale les objectifs de la communauté « en permettant aux nouveaux membres de cerner en préalable ces objectifs ainsi que les moyens mis en œuvre pour les atteindre » (Paul Muller, 2004, p54). Chaque membre est donc choisi à travers son adéquation avec les normes et les valeurs de la communauté.

---

[40] « Un ensembles de gens qui partagent des activités et ou des croyances communes et qui sont principalement liées par les relations affectives, des valeurs communes, et ou l'intérêt accordé aux relations inter-personelles.» (Brint, 2001, p8)

En terme de production de connaissances, la communauté est définie comme « une structures caractérisée par une haute fréquence d'interactions ayant pour but la génération et la diffusion de connaissances » (Paul Muller, 2004, p 51).

Elle est formée selon Wenger (1998) ; Brown et Duguid (1991) « de groupes d'individus engagés mutuellement dans une pratique, un projet commun, en vue de développer des compétences » (Wenger, 1998).

Dans la même veine, Cohendet et Diani (2003a) définissent la communauté comme « une véritable unité active de compétence et de connaissance, permettant de rendre possible et d'asseoir la production, l'accumulation et la validation de nouvelles connaissances » Cohendet et Diani, 2003 a, p 3).

D'un point de vue organisationnel, Cohendet et Diani (2003a) expliquent qu'une communauté s'organise spontanément autour d'un espace commun. Elle se distingue, selon eux, des formes traditionnelles par « une architecture non hiérarchique, une organisation faiblement structurée et peu rigide fondée sur le principe d'adhésion volontaire des agents en fonction du partage d'un certain nombre de valeurs, de normes ou d'intérêts communs » (Cohender et Diani, 2003 a, p 3).

De ces définitions émerge les différentes caractéristiques d'une communauté : interactions sociales, maintiens de relations de natures différentes, culture, valeurs et de normes partagées, intérêts communs, espace commun, etc. Ces caractéristiques se distinguent d'une communauté à une autres. Nous proposons dans la partie suivante les différents types de communautés identifiés dans la littérature.

I.B. Typologie des communautés

La littérature propose une classification très variée des communautés. Selon Creplet et al. (2001) chaque communauté est caractérisée par des traits spécifiques qui déterminent la manière dont elle apprend et se développe « certaines d'entre elles sont plutôt orientées vers la création de la connaissance et d'autres vers l'action; certaines sont définies et contrôlées par des mécanismes hiérarchiques, d'autres sont plus autonomes ». (Creplet et al., 2001, p 12).

Cohendet et al. (2003b) proposent plusieurs éléments qui distinguent les différents types de communautés. D'après eux les communautés se distinguent selon « l'objectif qu'elles poursuivent, les agents qui les composent, leur activité cognitive dominante, leur règle de recrutement, leur manière de produire de la connaissance, leur mode

d'apprentissage dominant et ce qui maintient leur cohérence ». (Cohendet et al. 2003b, p 102-103).

Nous présenterons dans ce qui suit, les types qui sont plus en relation avec notre contexte de recherche.

I.B.1. Communauté virtuelle

La notion de communauté virtuelle a fait l'objet de plusieurs définitions. Elle se trouve également au cœur d'un débat sur sa réalité. Certains auteurs[41] (Tremblay, 1998 ; Thompson, 1995) montrent le paradoxe d'associer le qualificatif « virtuel » à la notion de communauté. D'autres tentent de définir cette notion en pourtant un regard latéral sur ce débat.

Le terme de « communauté virtuelle » est apparu pour la première fois dans l'ouvrage de Rheingold, The Virtual Community, en 1993. Pionnier dans l'étude des relations sociales au sein des réseaux informatiques, il est le premier à introduire la notion de communauté virtuelle, et la définit comme un groupement socioculturel qui émerge lorsqu'un nombre suffisant d'individus participe à des discussions publiques et des relations humaines se tissent au sein d'un cyberspace.

> « *Social aggregations that emerge from the net when enough people carry on... public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace.*[42]» (Rheingold, 1993, p 5)

C'est la notion d'espace, ou plus exactement d'espace virtuel, qui rassemble le plus de travaux sur la question.

Pour Carver (1999) une communauté virtuelle est un regroupement de personnes qui partagent un espace ou un environnement commun. C'est dans cet espace que ces personnes s'engagent et interagissent.

---

[41] Selon Tremblay (1998) « la notion de communauté de communauté renvoie au collectif fondé sur la proximité géographique et émotionnelle, et impliquant des interactions directes, concrètes, authentiques entre ses membres. En paradoxe, le virtuel renvoie à l'idée d'abstraction, d'illusion et de simulation. » (Cité par Proulx, 2000, p 101)

[42] « les communautés virtuelles sont des regroupement socioculturels qui émergent du réseau lorsqu'un nombre suffisant d'individus participent à ces discussions publiques pendant assez de temps en y mettant suffisamment de cœur pour que des réseaux de relations humaines se tissent au sein du cyberspace » (Rheingold, 1993, p 5).

> *«Virtual communities are about aggregating people are drawn to virtual communities because they provide an engaging environment in witch to connect with other people sometimes only once, but more often in an ongoing series of interactions.* » Carver (1999, p 116)

Selon Hagel et al. (1997) la communauté virtuelle est représentée par un espace virtuel au sein duquel des personnes génère de nouvelles idées (composantes) en communiquant et agissant collectivement.

> «*Virtual communities are computer-medated spaces where there is a potential for an integration of content and communication with an emphasis on member-generated content.* » (Hagel et Armstrong 1997, p 142)

De ces définitions émergent les caractéristiques fondamentales des communautés virtuelles :

- Un espace virtuel commun : un espace virtuel sert de localité commune, de support d'interactions et d'échange pour les membres de la communauté. Il représente le répertoire d'informations et de connaissances de la communauté ou mémoire de la communauté.
- La production de connaissances : la communauté virtuelle fourni un espace qui met en commun les compétences et les expériences de ses adhérents. Cette mise en commun génère de nouvelles connaissances.
- Les rapports entre les participants : les rapports entre les participants constituent la raison d'être de la communauté. Ils sont assurés par des interactions continues et le partage d'intérêts, d'expériences et de répertoires communs.

I.B.2. Communauté de pratique

Présentée par Lave et Wenger (1991), La notion de communauté de pratique a suscité de nombreuses interrogations, certains aspects qui lui sont attribués tels que le partage de pratique, d'intérêts, de compétences individuelles, etc. offrent une perspective utile à l'étude de notre concept de recherche

Lave et Wenger (1991) définissent une communauté de pratique comme un ensemble de relations que les membres établissent au travers de leurs interactions, leurs ressources partagées et leurs intérêts de développer leurs compétences.

> « *Community of practice defines as a system of relationships between people, activities, and the world; developing with time, and in relation to other*

tangential and overlapping communities of practice. » (Lave et Wenger, 1991, p 98)

Selon Wegner (1998) les communautés de pratiques regroupent des personnes, de façons informelles, pour partager une passion sur une pratique donnée ou résoudre des problèmes difficiles, et qui approfondissent leurs connaissances en s'engageant mutuellement dans cette pratique.

> « Members of a community practice are informally bound by what they do together –from engaging in lunchtime discussions to solving difficult problems- and by what they have learned through their mutual engagement in these activities.[43]» (Wenger, 1998, p 2).

Wenger (1998) rajoute que ces communautés sont auto organisées lorsqu'elles émergent de façon spontanée. Il s'intéresse également à l'évolution et au fonctionnement des communautés de pratiques et explique que ces communautés évoluent au travers les pratiques que ses membres développent :

> « *Any community of practice produces abstractions, tools, symbols, stories, terms, and concpets that reify something of that practice in a congealed form.* [44]» (Wenger, 1998, p 59)

Pour Wagner (2000), les communautés de pratiques fonctionnent sur la base de trois éléments :

- Elles émergent de façon spontanée à travers l'initiative des membres. Ces derniers interagissent, établissent des normes et s'engagent mutuellement autour de leurs pratiques.
- Ses membres sont liés par un sens de l'entreprise commune « *joint enterprise* ».
- Au travers de leurs pratiques, Ses membres développent un répertoire partagé de connaissances, d'expertises, de ressource, etc.

Cohendet et al (2003b) montrent que la longévité d'une communauté de pratique dépend du degré d'engagement et de l'implication de ses membres. Ses limites ne sont pas clairement définies elles sont fonction du niveau d'implication et de la centralité des membres qui la composent.

---

[43]« Les membres d'une communautés de pratiques sont d'une façon informelle liées par ce qu'ils font ensemble- par un engagement dans des discussions…, à résoudre des problèmes difficiles- et par ce qu'ils ont appris par leur engagement mutuel dans ces activités » (Wenger, 1998)

[44] Traduction

Dans ses travaux les plus récents, Wagner (2002) associe l'attribut « virtuel » à la notion de communauté de pratique pour décrire les communautés de pratique dont les membres utilisent un espace virtuel partagé comme support d'interaction. Foray (2004) reprend par la suite ce terme pour définir un espace virtuel comme le point de rencontre pour les experts et les professionnels afin qu'ils puissent échanger, partager, autours de pratiques.

I.B.3. Communauté épistémique

La communauté épistémique a fait l'objet de nombreux travaux aussi bien en épistémologie sociale qu'en économie de la connaissance (Schmitt, 1995; Cowan et al., 2000, Cohendet et al., 2003). Ces études recentrent notre attention sur la notion d'apprentissage et de création de savoir.

D'après Cowan, David et Foray (2000), une communauté épistémique revoie à un groupe de personnes qui collaborent dans le but de produire des connaissances :

> « *They are small groups of agents working on a commonly acknowledged subset of knowledge issues and who at the very least accept a commonly understood procedural authority as essential to the success of their knowledge activities.*» (Cowan, David et Foray, 2000, p 234).

D'autres travaux soulignent la dimension organisationnelle de la communauté épistémique. Elle se fonde, selon Haas (1992), sur l'existence de croyances communes, la présence de règles et de normes qui dirigent l'action de ses membres.

> « *An epistemic community is a network of professionals with recognized expertise and competence in a particular domain and an authoritative claim to policy-relevant knowledge within that domain or issue-area.* » (Haas, 1992, p 3)

Pour Cohendet et al. (2003b) une communauté épistémique se caractérise par la présence d'une autorité « procédurale » qui définit les objectifs de la communauté à travers un ensemble de critères, de règles et de codes de conduites. Les contributions (cognitives) sont évaluées et validées suivant les critères fixés par l'autorité procédurale :

> « *…la validation de l'activité cognitive d'un représentant de la communauté est faite suivant les critères fixés par l'autorité procédurale. L'objet de l'évaluation concerne la contribution individuelle à l'effort vers le but collectif à atteindre....la règle de recrutement est ainsi définie relativement à la*

contribution qu'un représentant effectue pour réaliser son but .» (Cohendet et al, 2003b, p 106).

Récemment, le concept de communauté épistémique est employé pour décrire la forme d'une activité de cognition distribuée (Roth, 2008). Ce qui rapproche ce concept de celui des communautés Open Source.

I.C. Apports et limites de la littérature sur les communautés Open Source

Comme nous l'avons introduit dans le premier chapitre (II.B.1), la littérature identifie différentes caractéristiques des communautés Open Source. En se basant sur ces travaux, une communauté Open Source se distingue par:

- une activité effectuée exclusivement dans un environnement virtuel et qui intègre des milliers de bénévoles distribués (Dalle et al. 2005)
- un régime de propriété intellectuel qui protège le droit de créer et diffuser de manière continue et libre des connaissances dans l'objectif de développer des produits logiciels techniquement complexes (Stallman, 1999),
- un environnement favorable à la création de connaissances et à l'amélioration de compétences (Cohendet et al, 2003b),
- et une organisation du travail peu structurée (Crowston et al, 2005).

Nous pouvons constater que les caractéristiques de l'Open Source présentent des propriétés communes avec les concepts étudiés. Dans ce qui suit, nous présentons en premier lieu les travaux qui étudient les communautés Open Source. Nous dégageons par la suite les apports et limites des concepts étudiés pour caractériser les communautés Open Source.

I.C.1. Les spécificités des communautés Open Source

Dans cette partie nous présentons les travaux identifiant les caractéristiques des communautés Open Source. Nous présentons, tout d'abord le caractère distribué de ces communautés. Nous décrivons, ensuite leurs avantages en termes d'apprentissage. Nous exposons, enfin les dimensions organisationnelles qui distinguent les communautés Open Source. Ce dernier point fera l'objet de la prochaine section.

I.C.1.a. Le caractère distribué des communautés Open Source

Les communautés Open Source sont souvent considérées comme un ensemble d'acteurs indépendants et volontaires, qui utilisent des ressources complémentaires pour produire des logiciels. Comme nous l'avons introduit dans le premier chapitre (cf. Section 2), de nombreux travaux mettent l'accent sur la nature distribuée de ces communautés. D'après eux, l'ouverture du code source, rassemble un très grand nombre de participants. Ce nombre dépasse des fois les milliers de développeurs. En plus de leurs nombres importants, ces participants sont dispersés géographiquement et utilisent Internet pour communiquer et coordonner leurs actions. Les Mailing lists, les systèmes de contrôle de code source (CVS) et les supports de gestion de bogues (Bug repository) représentent des outils qui supportent l'activité de la communauté. Von krogh et al. (2003) montrent que la distribution concerne également l'activité puisque la conception du logiciel est réalisée par plusieurs acteurs travaillant individuellement sur une tâche précise et rendue disponible aux autres.

La distribution est vue par Conein (2003) dans l'acquisition de la connaissance. Il montre dans la communauté Open Source *Debain*[45], la connaissance nouvelle n'est pas acquise de façon individuelle mais par référence à une communauté d'expertises collectives, développés sous un mode distribuée.

> « *Les communautés du logiciel libre se présentent donc plus comme des communautés d'expertise collective que comme des communautés d'assistance aux utilisateurs. La communauté francophone des debianistes rassemble des hackers engagés à la fois dans des processus d'exploration et d'apprentissage. L'expertise collective sous un mode distribué passe par une combinaison originale entre exploration et assistance où l'innovation est valorisée en déstabilisant les connaissances.* » Conein (2003, p 12)

[45] Debain est une organisation « *Composée uniquement de bénévoles, dont le but est de développer le logiciel libre et de promouvoir les idéaux de la Free Software Fondation. Le projet Debian a démarré en 1993, quand Ian Murdock invita tous les développeurs de logiciels à participer à la création d'une distribution logicielle, compète et cohérente, basée sur le nouveau noyau Linux. Le projet Debian a publié un certain nombre de documents qui mettent en évidence ses valeurs et expliquent ce que signifie être un développeur Debian* » (Pons et al., 2006, p. 14).

I.C.1.b. L'apprentissage dans le cadre du développement Open Source

Les communautés Open Source sont largement considérées comme des structures favorables aux partages, à la création de connaissances et particulièrement à l'apprentissage.

Pour Von Hippel et Von krogh (2003), faire partie d'une communauté Open Source, présente des avantages en termes d'apprentissage. En étudiant le projet Freenet[46], ils montrent que l'implication des développeurs dans le projet leurs permet de partager des ressources communes et d'apprendre de la communauté.

Dans la même veine, Foray et Zimmerman (2002) expliquent que la complexité liée au développement d'un logiciel nécessite de l'expérimentation ce qui engendre des effets très importants d'apprentissage par l'usage :

> «*Le développement de logiciel libre peut être compris comme un processus d'apprentissage … le fait de donner à l'utilisateur l'accès au code source permet d'exploiter au mieux une très grande intelligence distribuée : les millions d'usagers qui révèlent des problèmes et les milliers de programmeurs qui trouvent comment les éliminer.*» (Foray et al. 2002, p 5).

En étudiant la communauté Linux[47], Cohendet et al. (2003b) identifient deux types d'apprentissage: l'apprentissage par interactions et l'apprentissage par essais-erreurs. Selon Cohendet et al, ces deux types d'apprentissage sont dûs à l'usage intensif d'internet et au partage d'un langage ou répertoire commun.

[46] Freenet est un logiciel libre développé par les fondateurs du réseau freenet. Il permet de communiquer selon protocole du réseau et se fonde sur le même principe que Napster, à savoir, la capacité de communiquer des données d'ordinateur à ordinateur (peer-to-peer).

[47] Le noyau de système d'exploitation Linux est né en 1991 « *grâce à un étudiant de l'université d'Helsinki. La réussite du nouveau système devra son salut à l'idée de son créateur, L. Torvalds, d'inscrire son projet sous les termes de la licence GPL et de proposer à tous les programmeurs et autres hakers (les premiers programmeurs sur les systèmes Unix devenus de véritables « gourous » dans leur domaine) d'Internet de l'aider.* » (Pons et al., 2006, p. 13). La création de ce noyau a permis l'apparition de distributions Linux gratuites, constituées du noyau et de nombreux autres logiciels libres. Les distributions destinées au grand public les plus connues sont Debain, Mandriva Linux, etc.

I.C.1.c. L'organisation des communautés Open Source

Les questions sur l'organisation des communautés Open Source ont fait l'objet d'un nombre important de travaux. Elles se retrouvent également au cœur de notre problématique. Nous les introduisons dans cette partie en mettant en avant les aspects qui caractérisent les communautés Open Source.

Au départ, la littérature identifie les communautés Open source comme des structures « anarchique », décentralisé et non régulées, où tout le monde peut y contribuer librement sans contraintes (Raymond, 1999 ; Hertel et al., 2003). Cette approche est par la suite remise en question par d'autres auteurs (O'Mahony et al., 2007) ; Crowston et al., 2005; Kogut et Metiu, 2000) qui conteste la nature anarchique et complètement décentralisée des communautés Open Source. Ces auteurs démontrent que les communautés Open Source s'organisent par la présence de hiérarchie (Crowston et Scozzi, 2005 ; Kogut et Metiu, 2000) ou de leadership (O'Mahony et Ferraro, 2007). Le rôle du leadership dans la communauté est d'évaluer, contrôler, valider ou rejeter les contributions des participants. Nous développons ces notions dans la section suivante.

Pour Sharma et al. 2002, les communautés Open Source sont organisées sur la base de deux mécanismes :

- un mécanisme de réciprocité : les communautés Open Source représentent des lieux d'échanges dans lesquels les gents signalent les bugs et attendent que d'autre développeurs le résolve. De la même façons les membres qui contribuent à la résolution de problème attendent que les autres contribuent à d'autre partie du projet. Ce mécanisme permet de maintenir les échanges entre les participants et contribue par conséquent à l'organisation de l'activité (Moon et al., 2000) ;
- un mécanisme de réputation et de reconnaissance des pairs : la capacité de coordonner les compétences provient du fait que les contributeurs dans un projet sont informés des experts capable de résoudre un problème particulier. La coordination se base donc sur la réputation et la reconnaissance des pairs (Lerner et Tirole, 2000). C'est ce que Benkler (2002) appelle aussi la« production par des pair ». Selon Benkler (2002), le fonctionnement des communautés Open Source repose sur l'action individuelle et le principe de l'évaluation par les pairs. Ces actions proviennent des initiatives individuelles et sont organisées sur la base de la reconnaissance des pairs.

I.C.2. Rapprochements avec les concepts étudiés

En partant des éléments issus de la littérature caractérisant les communautés Open Source, nous développons dans ce qui suit les rapprochements avec les concepts étudiés, à savoir les communautés virtuelles, de pratiques et épistémiques.

I.C.2.a. Rapprochement avec les communautés virtuelles

La nature virtuelle des communautés Open Source s'observe dans l'usage d'Internet comme principal outils de communication. En effet, l'utilisation exclusive de ces moyens de communication permet la création d'espaces de travail virtuel. C'est dans cet espace que les membres d'une communauté Open Source se rassemblent de manière continue et s'engagent autour d'une activité commune. Cet espace constitue également un lieu de partages dans lequel des relations se tissent. Nous retrouvons ici les caractéristiques fondamentales des communautés virtuelles et rejoignons les travaux (Crowston et Scozzi, 2005) qui considèrent que les communautés Open Source, peuvent être analysées comme des organisations virtuelles.

I.C.2.b. Rapprochement avec les communautés de pratique

Certains travaux rapprochent les communautés Open Source à la notion de communauté de pratique virtuelle développée par Foray (2004) et bien avant lui Wagner (2002). Bien qu'à travers cette notion, une communauté de pratique est définie par les relations que les membres établissent au travers de leurs interactions dans espace virtuel partagé, leurs intérêts communs et leurs échanges de connaissances continues; Elle se développe, cependant, autour d'une pratique partagée dans un même métier contrairement à une communauté Open Source qui peut regrouper des individus ayant des métiers différents. De plus, comme nous l'avons explicité dans le paragraphe (I.B.2), les communautés de pratique peuvent être auto-organisées lorsqu'elles émergent de façon spontanée à travers l'initiative des individus qui la compose (Wenger, 1998). Il n y a donc pas de hiérarchie dans les communautés de pratique (Brown et al., 1991) alors que comme nous venons de l'expliciter l'organisation des communautés Open Source sont régulées par des mécanismes hiérarchiques.

I.C.2.c. Rapprochement avec les communautés épistémiques

Le concept de communauté épistémique est également employé pour décrire la forme du développement Open Source. Pour analyser ce rapprochement nous présentons les résultas de l'étude de Cohendet et al. (2003b) sur la communauté Linux.

Cohendet et al. (2003b) démontrent que Linux est une forme hybride entre communauté de pratique et épistémique. Selon les auteurs, l'objectif principal de Linux, au départ, est orienté vers la création de connaissances dans une pratique bien déterminée. La connaissance est ainsi produite dans une pratique bien déterminée, une pratique qui « *s'appuie sur des bases de connaissances de différentes nature* » (Cohendet et al., 2003b, p 114). Ces constations rapproche la communauté Linux plus des communautés épistémiques. Un autre point commun avec une communauté épistémique que Cohendet et al. (2003b) observent dans Linux est la présence d'une autorité procédurale :

> « *...placé à part de la majorité des développeurs, une autorité procédurale a été créée a fin d'évaluer les apports de chacun et gérer l'accroissement des flux de contributions* » (Cohendet et al., 2003b, p 113).

Pour résumer, la communauté Open Source peut être considérée comme une forme d'organisation particulière impliquant des développeurs de divers domaines et compétences dans une activité collective. Elle constitue un espace virtuel d'interactions et d'apprentissage et se distingue principalement par la distribution géographique des membres qui la composent et par un mode d'organisation qui diffère des modes traditionnels[48]. Nous retenons ainsi les caractères épistémiques et virtuels des communautés Open Source. Et nous nous adhérons aux approches qui considèrent le développement Open Source comme champ d'application croissant du concept communautés intensive en connaissances (Dalle et Jullien, 2003, Kogut et Metiu, 2001, Lerner et Tirole, 2000) et plus spécifiquement comme « *un mode coopératifs de production de Connaissances parmi les membres d'une communauté épistémique distribuée.* » (Dalle et al., 2005, p. 397).

[48] Nous faisons ici références à la hiérarchie et le marché comme mode régulateur.

Section 2. L'apport de la littérature aux mécanismes de régulation des communautés Open Source

La revue de la littérature relative à l'organisation des communautés Open Source montre un paradoxe entre les travaux qui considèrent que ces communautés sont organisées anarchiquement et ceux qui soutiennent que les communautés Open Source apparaissent plus structurées.

Nous abordons dans cette section la question qui concerne la régulation des communautés Open Source. Pour traiter cette question, nous introduisons cette section par une définition de la notion de hiérarchie : le mécanisme de régulation traditionnel. Nous soulignons par la suite l'aspect controversé de l'organisation des communautés Open Source et leur rapport à cette notion.

La notion de hiérarchie est apparue suite aux interrogations sur la nature de la firme. Ceci a travers la vision qu'apporte la théorie des coûts de transactions (Willamson, 1975). En effet, dans la lignée des travaux de Coase (1937), Willamson (1975) met en avant le rôle de la hiérarchie comme « *mode alternatif de support des transactions* ». Dans ce cadre, la hiérarchie désigne le recours à la subordination plus tôt qu'aux mécanismes du prix inhérent au marché.

La hiérarchie est définie selon Mintzberg (1979) comme le moyen de coordonner toutes les décisions dans l'organisation. Elle accorde aux dirigeants le pouvoir de décider et de contrôler par supervisions directes la bonne application de leurs décisions.

Selon ces visions de la hiérarchie, l'organisation se fait par des directives, des ordres et des relations de subordinations verticales. Pour certains auteurs cette vision classique n'est plus en mesure d'influencer le fonctionnement des communautés Open Source. D'après eux, ces communautés émergent, comme des modes d'organisation anarchique. Pour d'autre, l'absence de hiérarchie formelle ne signifie pas que ces communautés sont dépourvues de structuration et de procédures d'autorité et impliquent même des modes d'organisations radicalement nouveaux. Nous exposons tout au long de cette section ces différentes approches de l'organisation des communautés Open Source.

II.A. La vision anarchique de l'Open Source

Eric Raymond (1998) est le premier à avoir décrit la manière dont une communauté Open Source s'organisait. Dans son article intitulé «*The Cathedral and the Bazaar* », il

présente cette communauté comme une structure de production auto-organisée, spontanée et efficiente. Raymond (1998) explique l'efficacité du développement Open Source par la capacité de mettre en commun des compétences venant de domaines différents. Selon Raymond (1998), cette efficacité exige une structure fortement décentralisées, un bazar, où les membres sont des négociants marchands qui décident de manière autonome quand et comment contribuer à la communauté. Il considère ainsi l'organisation des communautés Open Source comme anarchique, en référence à un modèle de « bazar », fondée sur une très forte décentralisation auprès de contributeurs non sélectionnés et dépourvus de coordination formelle. Il n'y a donc pas de structure hiérarchique qui définit les tâches et les attribuent à des acteurs en fonction de leurs statuts comme pour le cas des structures traditionnelles. De la sorte, l'organisation des communautés Open Source se fait de manière spontanée, comme dans un « bazar ».

Les travaux de Raymond ont par la suite inspiré grand nombre d'auteurs (Demil et al., 2005; Elliott et al., 2003) qui défendaient l'idée que les communautés fonctionnent sans un mode de coordination hiérarchique entre les acteurs et à la manière du « bazar ». L'approche de Raymond a cependant fait l'objet de beaucoup de critiques. Bezroukov (1999) montre, par exemple que cette approche néglige le rôle de la hiérarchie ou du contrôle du « leader de projet » dans l'encadrement de l'activité et la gestion de conflits :

> « *The bazaar metaphor disregards important aspects of FLOSS development processes, such as the importance of project leader control, the existence of de-facto hierarchies and emergent leadership, the danger of information overload and burnout, and the possibility of conflicts that cause a loss of interest in a project or forking.*» (Bezroukov, 1999, p3)

II.B. La vision hiérarchique de l'Open Source

La conception de mode d'organisation anarchique, décentralisée et informel, parait être compatible avec les nouvelles formes qui favorisent l'émergence et la croissance de connaissances, en multipliant les interactions entre plusieurs acteurs. Néanmoins, organiser des contributions diverses nécessite un minimum d'ordre. En effet, plusieurs observations remettent en question la nature très décentralisée des communautés Open Source, et montrent, qu'en réalité, ces communautés sont marquées par l'existence de hiérarchie « informelle ». Ces observations émergent d'un nombre important de travaux.

Le premier travail qui a observé l'émergence de hiérarchie dans les communautés Open Source est celui de Cox (1998). Dans son étude sur le projet Linux, Cox (1998) montre que l'organisation du travail dans ce projet n'est pas totalement démocratique. Il révèle que l'accès aux échanges n'est pas accordé à tous les participants. En effet, cet accès est réservé à une catégorie de développeurs qui ont la liberté d'intégrer ou pas d'autres participants dans l'échange. La structure de communication résultante est hiérarchique. Selon Cox, la hiérarchisation de la communication n'est pas conforme aux principes de l'Open Source selon lesquels un projet doit stimuler et autoriser la discussion générale. Ghosh et al. (2000) ont également souligné que la communauté Open Source est un ensemble organisé qui repose sur un centre stratégique la FSF : Free Software Foundation. Ce centre stratégique est à l'origine de plus de 17% de l'ensemble des projets développés par la communauté. Le modèle de développement des logiciels libres est donc mené par un nombre limité d'acteurs plutôt qu'un « bazar » constitué de nombreux utilisateurs/développeurs impliqués chacun dans de nombreux projets sans coordination centralisée.

Mockus et al. (2000) ont par la suite appuyé cette thèse. Ils observent, en étudiant le projet Apache[49], que l'écriture du code est réservée a un nombre limité de développeurs. En effet, seulement 15 développeurs contribuent en majorité (80%) à l'écriture du code, la conception et la validation de solution. Selon les auteurs, la contribution des autres participants est moins significative puisque leurs tâches consistent à déclarer les problèmes rencontrés ou de tester les solutions conçues par les développeurs :

> «*In successful open source developments, a group larger by an order of magnitude than the core will repair defects, and a yet larger group (by another order of magnitude) will report problems* » (Mockus et al. 2000, p 9).

Pareillement, Crowston et Howison (2005) ont soutenu l'idée que le développement de logiciels libres était réalisé sur la base d'une structure sociale. Ils ont même développé un modèle qu'ils nomment *l'oignon*. Les auteurs présentent la structure des communautés Open Source en niveaux, formant un oignon, chaque niveau est composé d'équipes ayant un rôle spécifique dans le développement. Le modèle distingue quatre niveaux (cf. figure) :

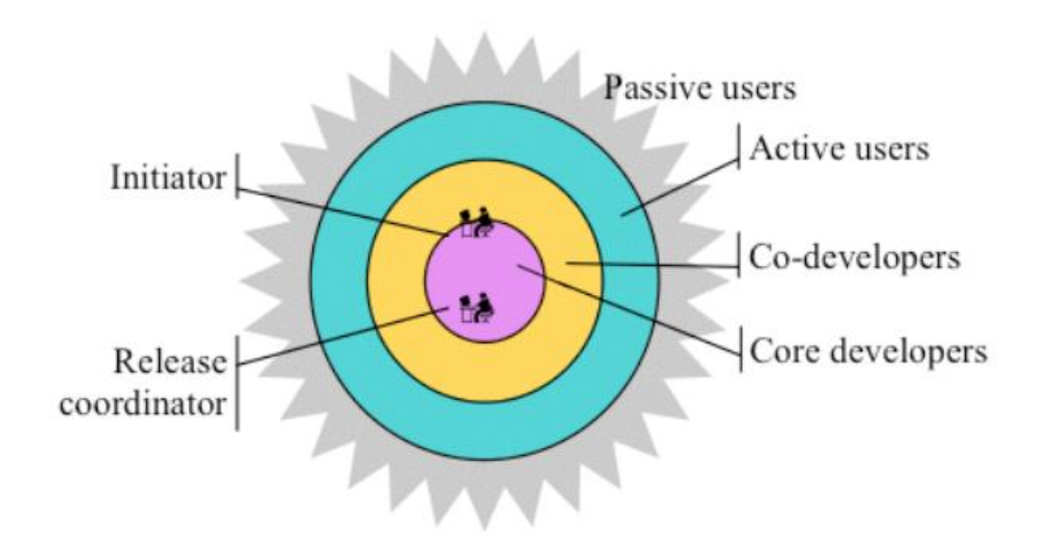
---

[49] Apache HTTP Server, souvent appelé Apache est l'appellation du logiciel Apache HTTP Server, produit par l'Apache Software Fondation. C'est un logiciel livre avec un type spécifique de licence, nommée licence Apache.

- au centre de l'oignon nous retrouvons le groupe des développeurs du cœur : les « *core developers* ». Le rôle de ces développeurs c'est de contribuer en majorité à l'écriture du code, de surveiller le design et l'évolution du projet. Ils représentent les développeurs les plus compétents qui ont le pouvoir le plus important dans la communauté. Les observations de Mockus et al. (2002) montrent que ces équipes sont généralement composées d'environ une douzaine de développeurs ;
- une deuxième couche est composée d'un nombre plus important de participants ou de « *co-développeurs* ». Ces derniers contribuent en majorité à la résolution de problèmes rencontrés. Leurs rôles consistent à déclarer de problèmes, concevoir ponctuellement de solutions et fixer les problèmes ;
- Nous retrouvons en troisième couche les utilisateurs qui participent activement au développement ou les « *active users* ». Ces derniers ne contribuent pas à l'écriture du code, mais participent occasionnellement à l'exécution de tâches simples ;
- La dernière couche représente les utilisateurs passifs ou les « *Passive users* », les moins impliqués dans l'activité de développement. Ces utilisateurs sont considérés comme de simples observateurs (Sproull et al., 2005).

Figure 2.1 : La structure des communautés Open Source,

Crowston et Howison (2005)



L'analyse de la littérature sur la structure organisationnelle des communautés Open Source montre un paradoxe. Au départ, en se basant sur les travaux de Raymond (1998) les communautés de logiciels libres sont considérées comme des entités entièrement décentralisées (un Bazar). Ensuite, d'autres travaux soulignent que ces communautés reposent sur une hiérarchie.

Elle émerge selon eux par la centralisation de la communication (Cox, 1998 ; Moon et Sprool, 2000), par la centralisation dans l'écriture du code (Cox, 1998; Mockus, 2000), et enfin par les statuts (Crowston et al., 2003).

Le modèle de l'oignon a permis de décrire les différents rôles dans une communauté de développement reflétant une structure hiérarchique. Cependant, le modèle présente seulement une image statique.

II.C. Intégration de nouveaux membres et identification des rôles

Nous venons de voir que la littérature s'est focalisée sur l'étude de la manière dont les communautés Open Source s'organisaient. Ces études observent que ces communautés se distinguent par leurs structures de gouvernance et l'existence de hiérarchie. Des structures qui font apparaître plusieurs rôles (cf. modèle de l'oignon (Crowston et Scozzi, 2005)). Nous tentons dans cette partie de comprendre comment ces rôles sont définis en l'absence de hiérarchie formelle. De récent travaux (Bird et al 2007, Herraiz et al. 2006 ; Von Krogh et al. 2003) postulent que pour intégrer une communauté Open Source et avoir un rôle, les nouveaux participant doivent suivre un processus ou une trajectoire d'intégration : « *process of joining* ». Les auteurs observent que la trajectoire d'intégration varie d'un projet à un autre. Selon eux, l'intégration d'un nouveau membre part de sa motivation de rejoindre la communauté. La motivation des nouveaux participants doit être perçue par les membres de la communauté pour qu'il soi intégrer définitivement dans la communauté.

II.C.1. La motivation des développeurs

Nous abordons dans cette partie la problématique de la motivation des développeurs, comme une dimension importante qui va nous permettre de mieux comprendre les aspects organisationnels et structurels des communautés Open Source.

Les motivations des individus est un thème de recherche majeur dans la littérature de l'Open Source (Dalle et al., 2005; Von Krogh et al., 2003 ; Lakhani et al., 2003; Lerner et al., 2001). Cependant, nombreux travaux divergent sur la nature des motivations impliquant les participants aux projets Open Source. En se référant aux travaux de Lakhani et Wolf (2003), ces motivations peuvent être de nature intrinsèque et extrinsèque.

II.C.1.a Les motivations extrinsèques

Pour Lakhani et Von Hippel (2003), la motivation de rejoindre une communauté Open Source provient d'abord de la volonté des participants de satisfaire un besoin particulier : « *en participant à ces communautés, les développeurs-utisateurs créent à des coûts quasi nul des logiciels qui leurs sont utiles et qui correspondent au mieux à leurs besoins*». D'après ces auteurs, les individus contribuent aux projets car ils sont sur de percevoir des bénéfices de leurs participations. Ces bénéfices peuvent êtres immédiats et/ ou futurs.

Les bénéfices immédiats se présentent sous forme d'un salaire ou le besoin d'un logiciel particulier (Lakhani et Von Hippel, 2003). En effet, la possibilité de la participation contre un salaire ne devrait pas être ignorée. Bon nombre de sociétés embauchent des programmeurs pour participer aux projets Open Source, afin de bénéficier de compétences de haut niveau à moindres coûts et délais, pour satisfaire au mieux leurs besoins spécifiques.

Les bénéfices futurs sont décrits sous forme d'avancement de carrière ou bien de développement de compétences. D'après Julien (1999), les participants peuvent améliorer leurs compétences de programmation en interagissant avec des développeurs de haut niveau, ayant des méthodes et des styles de programmation différents (Julien, 1999). De plus, Lahkani et Von Hippel (2003) montrent, qu'en contribuant à l'écriture du code, les développeurs construisent une réputation sur le marché du travail, et ceci, en faisant valoir aux employeurs potentiels, leurs compétences et talents de programmation. Selon ces auteurs, dans le cadre du développement Open Source, les compétences individuelles sont facilement observées à l'intérieur de la communauté (par les membres de la communauté) et à l'extérieur de la communauté (par les recruteurs potentiel).

II.C.1.b Les motivations intrinsèques des développeurs

Contrairement aux arguments précédents, Stallman (1999) soutient l'idée que les raisons qui motivent les développeurs à rejoindre les communautés Open Source ne sont pas financières. Selon Stallman (1999), les libertés d'utiliser, d'améliorer et de modifier le programme, motivent les individus à adhérer au développement Open Source. Il montre également, que les participants sont incités par la volonté de s'identifier à une culture communautaire («Hacker Culture ») qui est celle du partage.

Pour Torvalds et Diamond (2001), la participation des individus à l'Open Source s'explique par le plaisir de programmer : « le fait de participer aux développements de projets Open Source présente un plaisir, « fun » pour les développeurs » (Torvalds et Diamond, 2001). Selon Lakhani et Wolf (2003) c'est d'avantage une occasion qui permet aux participants de se montrer créatifs, et capables de relever des défis (Lakhani et Wolf, 2003).

Dans leur enquête portant sur près de 700 développeurs, Lakhani & Wolf (2003) observent que la liberté de création permise par l'Open Source est l'argument le plus souvent avancé par les personnes interrogées. Ceci montre que les motivations intrinsèques jouent un rôle prépondérant dans la décision des développeurs. Dans le même temps, ils montrent les motivations extrinsèques sont également présentes chez les individus. Autrement dit, la contribution et la participation des individus dans l'Open Source sont guidées par les motivations hétérogènes même si des motivations intrinsèques dominent.

> « *A clear finding from the cluster analysis is that the FLOSS community has heterogeneous in motives to participate and contribute. Individuals may join for a variety of reasons, and no one reason tends to dominate the community or cause people to make distinct choices in beliefs*. » (Lakhani et Wolf, 2003, p 14).

Cette étude indique en outre, que la nature de la motivation du participant n'a pas d'impact sur le degré de son implication dans le projet Open Source. En effet, Lakhani & Wolf (2003) montrent que lorsque l'on mesure le niveau d'engagement et le niveau d'effort, il n'existe pas de différences notables entre ceux qui sont rémunérés et ceux qui contribuent bénévolement à l'Open Source.

La catégorisation proposée par Lakhani et Wolf (2003) est extrêmement utile, pour donner une structure au jeu des motivations des développeurs. Dalle et al.(2004)

proposent par la suite une autre catégorisation de la motivation, appelée : «*motivation à la marge* ». Cette catégorisation met en évidence l'évolution des motivations des développeurs durant le projet. Gott et al.(2002) a aussi démontré en analysant les réponses de leur enquête FLOSS-2002 Survey, que les motivations se développent avec l'expérience. En effet, les résultats de l'enquête indiquent que pour plus de la moitié des personnes interrogées, les raisons initiales de participer aux développements étaient trop diffuses, ces raisons évoluant tout au long de la participation, vers un engagement « idéologique » à la communauté.

II.C.2.L'intégration des membres dans les communautés Open Source

Dans la partie précédente nous avons défini les raisons qui motivent les individus à rejoindre volontairement les communautés Open Source. Cependant, le développement d'un logiciel est une activité complexe. En plus d'être motivé, intégrer une communauté exige donc un très haut niveau de compétences. De plus, intégrer l'équipe de développement d'une communauté, exige que les compétences exposées par les contributeurs soient reconnues par les autres membres de la communauté.

La littérature montre que l'intégration d'un participant dans l'activité de développement d'une communauté Open Source suit une trajectoire ou un processus appelée « *process of joing* ».

En étudiant le projet Freenet, Von Krogh et al. (2003), observent que pour intégrer l'équipe de développement (avoir l'accès pour modifier le code source), le participant doit atteindre un certain niveau d'activité. Selon eux, le niveau d'activité représente l'intensité des efforts fournis par le participant. Il est déterminé à partir du nombre de messages postés, du nombre de solutions conçues, du niveau de priorité accordé au problème dont il est l'auteur, etc. Leurs résultats indiquent que la période d'intégration d'un nouveau contributeur, est comprise entre deux semaines et plusieurs mois Von Krogh et al. (2003) montrent que l'intégration des participants dépend également de la nature de leurs contributions. Ils observent que 40% des messages postés par les « Joiners »[50], sont des messages techniques. Ces contributions « techniques »[51], reflètent

[50] Les participants qui ont intégré l'équipe de développement.

[51] Ecriture du code, une contribution technique pour le dessin et l'implémentation du logiciel.

les compétences et les connaissances techniques du participant, et impliquent par conséquent son adhésion dans la communauté de développeurs :

> « *Joiners would more frequently report technical bugs in the software than other contributors, making developers aware of important "construction sites" in the project.* » (Von Krogh et al., 2003, p. 1237)

D'après Ducheneaut (2005), l'intégration d'un participant dans l'équipe de développement d'un projet Open Source dépend non seulement de ses compétences mais également de la réputation acquise dans le cadre d'autres projets.

Selon Von Krogh et al. (2003), l'intégration des contributeurs dépend de la complexité, la modularité et la transparence du projet :

> «*..the first development activity by immigrants is strongly determined by modularity, complexity, etc. of the target file or class* ». (Von Krogh et al. 2003, p 1239).

Les auteurs expliquent que la transparence du projet permet aux nouveaux venues de s'informer sur les objectifs de la communauté et de répondre par la suite à leurs attentes. Ce qui facilite, par conséquent leurs d'intégration.

II.C.2.a Les facteurs d'intégration

L'examen de la littérature nous amène à constater que l'intégration d'un nouveau contributeur à la communauté de développement, dépend de trois principaux facteurs à savoir l'engagement technique du contributeur, son niveau de compétences, sa réputation et la structure du projet qu'il souhaite intégrer.

L'engagement technique des participants

L'engagement technique d'un participant est représenté par l'intensité de ses efforts (conception de solutions, révision de solutions, écriture de code, déclaration de problèmes, ect..). Selon Bird et al.(2007), l'engagement du développeur décroit avec la durée de sa participation au projet. Ils montrent que l'admission des contributeurs en tant que développeurs nécessite au début de sa participation beaucoup d'efforts techniques. La contribution est donc plus importante au début (nombre de mails, nombre de patchs), elle diminue lorsque le participant intègre définitivement la communauté des développeurs. Selon Bird et al (2007), en exposant leurs compétences

et connaissances, les participants techniquement engagés sont des développeurs potentiels.

Le niveau de compétence et la réputation

Ducheneaut (2005) montre que les compétences du participant est un facteur qui détermine son rôle dans la communauté. Selon Bird et al. (2007), lorsque le développeur passe plus de temps sur le même projet, il acquiert progressivement des connaissances, en interagissant fréquemment avec les développeurs de la communauté. Cet effet d'apprentissage lui permet de devenir plus compétant et facilite par la suite son intégration dans la communauté. Bird, et al. (2007) observent également que le temps[52] nécessaire pour apprendre de la communauté, varie entre les trois projets qu'ils étudient : les projets Postgres, Python et Apache. Ils expliquent que ces variations peuvent être attribuées aux différences au niveau des normes institutionnelles relatives à chaque projet, des complexités techniques et des mécanismes de gouvernance.

Un deuxième facteur important de motivation est la réputation en effet, selon Lakhani et Wolf (2003) et Duncheneaut (2005), l'intégration d'un nouveau participant dans l'équipe de développement dépend de sa réputation. En se basant sur l'étude des communautés Postgres, Python et Apache, Bird et al. (2007) proposent une modélisation de l'intégration dans les communautés Open Source. Cette modélisation révèle que l'intégration d'un participant varie selon trois facteurs : le premier est la durée qu'il a passé dans la communauté, le deuxième est le nombre de patchs qu'il a soumit et le troisième est la réputation qu'il a pu construire au cours de son engagement dans la communauté.

La structure de la communauté

Chaque communauté Open Source se caractérise par plusieurs aspects tels que les normes culturelles et des structures de gouvernances spécifiques. L'intégration d'un nouveau membre dépend de ces aspects. Pour certaine communauté comme Apache, la structure de gouvernance est formelle et hiérarchique. En effet, la communauté Apache a une structure bien élaborée qui donne une description détaillée des différents rôles dans la communauté. Les développeurs du centre ou les « core developpers » occupent

---

[52] Bird, et al. (2007) mesurent ce temps par la durée entre la première contribution dans la « mailing list », et la première conception de solution.

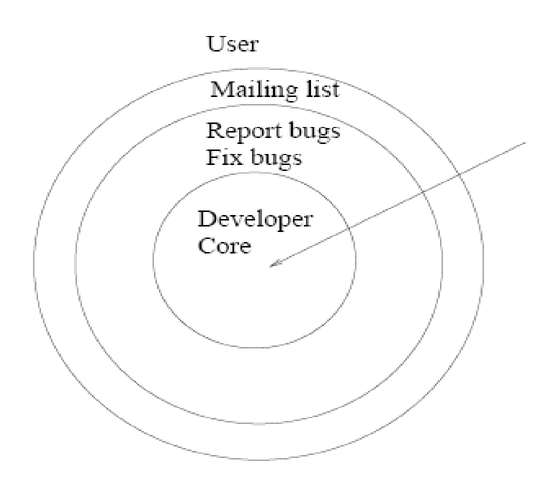
les rôles les plus importants et prennent toutes les décisions majeures dans le projet. Plus précisément, ce sont les seuls qui ont le droit de voter pour accorder un statut aux contributeurs et les intégrer dans l'activité de la communauté.

D'autres communautés, comme par exemple Postgres, ont une structure moins formelle. L'intégration d'un participant dans l'équipe de développement dépend de ses compétences ainsi que de son investissement dans le projet. Selon Bird et al. (2007), un participant intègre le noyau du projet Prostgres lorsqu' il devient si productif que les développeurs du coeur cèdent, et le laissent agir.

II.C.2.b Le processus d'intégration des nouveaux membres

En s'appuyant sur le modèle de l'oignon (Crowston et al., 2003), Herraiz et al. (2006) montrent que l'intégration d'un nouveau participant dans les communautés Open Source est un processus composé de plusieurs étapes (cf. figure 2.2).

Figure 2.2 : Le processus d'intégration des nouveaux membres selon le modèle de l'oignon (Herraiz et al. 2006)



Dans une première phase, les participants explorent l'activité de la communauté en visitant de temps en temps son site web. Ils sont ainsi de simples utilisateurs qui s'intéressent à l'activité de la communauté. Leurs objectifs est de collecter le plus d'informations sur la communauté (règles, spécificités techniques de ses projets, etc.). Dans une seconde phase, les participants sont en mesure d'interagir avec les membres de la communauté, de déclarer des problèmes et de soumettre des patchs.

Lorsque les contributions des participants deviennent plus significatives, la communauté leurs accorde l'accès au code source (la possibilité de modifier le code source). A travers cet accès le rôle des participants est important et plus influant dans la communauté. Si les participants collaborent dans des activités de haut niveau (techniquement complexe) durant une certaine période, ils intègrent définitivement le « Core group », c'est la dernière phase du processus d'intégration.

Au terme de cette seconde partie, nous pouvons retenir des différents concepts tirés de la littérature que l'intégration d'un nouveau membre dans le cadre du développement Open Source n'est pas automatique. En effet, intégrer une communauté ou un projet Open Source nécessite non seulement que le contributeur soit motivé mais également que ses efforts soient reconnus par l'ensemble des développeurs de la communauté. Son intégration suit une trajectoire et dépend de nombreux de facteurs tel que les compétences acquises la réputation du contributeur, la structure de gouvernance du projet ou de la communauté ou de sa complexité technique.

Section 3. Synthèse et apport aux questionnements

Au travers de cette revue nous pouvons constater qu'un nombre important de travaux a étudiés la nature et les caractéristiques des communautés Open Source. Différentes interprétations ont émergées. D'une part, les communautés Open Source ont été décrites comme une forme intégrant des milliers de bénévoles distribuée dont l'activité est effectuée dans un environnement virtuel favorable à la création de connaissance et à l'amélioration de compétences. D'autre part, la littérature montre que l'ouverture, caractéristique fondamentale du développement Open Source, ne signifie pas que le développement est réparti égalitairement entre les membres de la communauté.

En se basant sur ces développements, le Bazar, modèle développé par Raymond (1998) à été jugé inefficace pour décrire la structure des communautés Open Source. C'est dans ce sens que l'analyse des réseaux d'interactions au sein de plusieurs projets Open Source élaborée par Crowston et Howison (2006) montrent que les projets Open Source sont souvent hiérarchiques. Ils décrivent ainsi le modèle de l'oignon donnant une description générale de l'organisation des projets Open Source.

Dans une étude du noyau de Linux, Lee et Cole (2003) observent qu'un nombre limité de développeurs (le noyau de développeurs) possèdent le pouvoir de décider au sein du projet. Les taches les moins critiques sont quand a elle déléguées au membre périphérique. Mockus et al. (2002) proposent une structure des projets Apache et Mozilla composé d'un noyau, d'un groupe plus large que le noyau composé de membre dont la fonction consiste à corriger les problèmes et d'un groupe encore plus large qui déclarent et testent les problèmes.

De considérables attentions des chercheurs ont également été accordées à la question des motivations des participants dans les communautés Open Source. Les initial travaux de Lerner et Tirole (2002) sur ces questions s'interrogant sur les raisons qui incident les programmeurs à produire gratuitement un bien public. Les recherches empiriques qui ont suivi ont montré que les motivations des développeurs de participer aux communautés Open Source sont à la fois intrinsèques (Lakhani et Wolf, 2005; Von Hippel, 2001) ou extrinsèques (Ghosh et al., 2002).

Au-delà de ces questions de motivation, Von Krogh et al. (2003) quand a eux montrent qu'intégrer la communauté nécessite en plus d'être motiver, que le contributeur s'investi dans la vie active de la communauté en mettant en avant ses compétences ainsi que sont intérêt pour la communauté. Ainsi, ils décrivent une trajectoire qu'ils appellent « *joins scripts* » représentant des activités typiques dans lesquels les développeurs se livrent pour intégrer le noyau quand ils se déplacent de la périphérie au noyau.

Au terme de ce chapitre, nous pouvons dire que la littérature décrite sur la nature et l'organisation des communautés Open Source nous donne plus d'éclairage sur la structure et les caractéristiques de ces communautés. Néanmoins, moins d'attention a été accordée à la question de savoir comment l'effort des individus est mis à profit pour résoudre les problèmes distribués dans ces communautés (exceptions les travaux de Lee et Cole, 2003 et Von Krogh, Lakani et Spaeth, 2003). Plus spécifiquement, ***notre objectif est d'approfondir ces considérations empiriques pour tenter d'expliquer comment l'activité est coordonnée dans de tels contextes.***

CHAPITRE III-LA COORDINATION DANS LES COMMUNAUTES OPEN SOURCE

TABLE DES MATIERES

CHAPITRE III-LA COORDINATION DANS LES COMMUNAUTES OPEN SOURCE

Nous nous proposons dans cette section de traiter un autre niveau de l'organisation qui est la coordination. Nous étudions la coordination comme un mécanisme qui permet de concrétiser les dispositifs de régulation identifiés dans la section précédente. Nous présentons dans un premier temps les différentes approches de la coordination identifiée dans la littérature et nous verrons combien la coordination est primordiale pour le travail en communauté. Nous abordons ensuite les mécanismes de coordination mises en place dans les communautés Open Source.

Section1. Les différentes approches de la coordination

Plusieurs domaines se sont intéressés à la question de la coordination, tel que l'économie, la sociologie, les sciences des organisations ou l'informatique. Elle est l'objet de plusieurs définitions généralement similaires qui s'accordent sur le fait que la coordination est un facteur essentiel pour l'organisation d'une activité. Elle est encore plus pertinente dans un contexte tel que la communauté ou l'articulation, la synchronisation et la structuration de l'action collective sont des enjeux majeur pour garantir son efficacité.

I.A. Définition de la coordination

En science des organisations, Chandler (1962) propose une première définition de la coordination comme une fonction qui permet de structurer et de faciliter les transactions entre plusieurs acteurs interdépendants. Elle représente, selon Thompson (1967) l'ensemble des protocoles, tâches conçues pour relier des actions en provenance de plusieurs unités interdépendantes. Pour Lawrence et al. (1967) le rôle de la coordination est d'assurer l'interconnexion entre les différentes sous-unités d'une organisation.
Les travaux les plus remarquables dans cette discipline sont ceux de Mintzberg (1999). En effet, Mintzberg définit les mécanismes de coordinations permettant de gérer le

travail des salariées. Selon lui, la coordination repose sur la communication et le contrôle. Elle peut s'expliquer à partir de trois mécanismes (Mintzberg, 1999) :

- o La coordination par ajustement mutuel : repose sur la communication et les échanges entre acteurs. Le contrôle du travail n'est pas hiérarchique et la coordination repose sur l'autonomie et le savoir faire individuel. Elle est nécessaire pour gérer le travail dans des situations complexes car elle permet la réactivité et l'adaptation aux situations non programmées.
- o La coordination par supervision directe : Dans ce cas le contrôle du travail est effectué par un seul ou un petit groupe d'individus. Ce mécanisme de coordination repose sur la ligne hiérarchique et la centralisation du pouvoir au niveau de la hiérarchie.
- o la standardisation : elle est le signe d'une centralisation des pouvoirs techniques par la techno structure car la coordination est intégrée dès l'amont, dans le programme de conception du travail des analystes. La standardisation peut être de quatre ordres : une standardisation des procédés de travail (cadre la communication entre opérateurs et la supervision directe au moyen de règlements de méthodologies de travail qui définissent les tâches de chaque acteur), une standardisation des produits et des résultats (la définition des caractéristiques du produit et la coordination se fait au niveau de la technostructure), la standardisation des qualifications (les salariées sont affectés à un poste en fonction de leur qualifications, leurs savoir faire) et une standardisation des normes (repose sur les normes et les croyances partagées par tous les membres de l'organisation pour une harmonie d'ensemble).

Mintzberg (1982) explique que le choix d'un mécanisme et sa mise en place dépend de la taille de l'organisation. Il défini ainsi un cycle des mécanismes en fonction de la taille de des organisations (cf. figure 2.3). Selon Mintzberg (1982), dans les organisations de petite taille, les activités sont coordonnées en reposant sur les interactions entre acteurs c'est-à-dire par ajustement mutuel, avec son développement elle est contrainte de se structurer, un contrôle direct par la hiérarchie est nécessaire. Dans ce cas le mécanisme de coordination le plus appropriée est la supervision directe. A partir d'une certaine taille, l'entreprise ne peut plus coordonner toute l'activité par une supervision directe, pour contrôler l'activité elle doit donc « standardiser l'activité » ou se coordonner par ajustement mutuel.

Figure 2.3 : Le cycle des mécanismes de coordination (Mintezberg, 1982)

Ajustement mutuel —— **Supervision directe** —— **Standardisation** —— **Ajustement mutuel**

La coordination a également fait l'objet de définitions de la part de chercheurs en informatique et particulièrement les travaux sur les systèmes de coopération CSCW[53]. Singh et al. (1992) définissent *« la communication comme un mécanisme d'intégration, d'harmonisation et d'ajustement de travail individuel vers un objectif large ».[54]* Au sens de Reezigt (1995) la coordination consiste à « *établir l'harmonisation entre les tâches dans le but d'assurer que ces tâches, séparées, soient réalisée en temps prévu, dans de bon ordre et correctement* »[55].

Dans la même discipline de recherche Schmidt et Simone (1996) présentent un cadre conceptuel des mécanismes de coordination qui visent à fournir des bases pour concevoir des systèmes de travail coopératif distribuée. Ils définissent ainsi la coordination comme le travail d'articulation nécessaire pour gérer une activité indépendante, distribuée et complexe. Selon Schmidt et Simone (1996), un mécanisme de coordination possède deux composantes : un protocole de coordination qui regroupe un ensemble de procédures et de conventions révélant les moyens d'articulation d'activités indépendantes qui fournissent et un artefact[56] qui présente l'état d'exécution du protocole.

> « *A coordination mechanism is a construct consisting of a coordinative protocol (an integrated set of procedures and conventions stipulating the articulation of interdependent distributed activities) on the one hand and on the other hand an*

[53] Le Computer Supported Coopérative Work (Travail coopératif assisté par ordinateur, CSCW) est une communauté qui rassemble des chercheurs issus de disciplines diverses (psychologie, linguistique, sociologie, informatiques, etc.) dont le but d'analyser et de concevoir des machines à coopérer.

[54] Définition citée par Malone et crowston (1994): « *The integration and harmonious adjustment of individual work efforts towards the accomplishment of a larger goal*.» (Singh & Rein, 1992 par Malone et crowston, 1994, p 92)

[55] Définition citée par Malone et crowston (1994): « Establishing attunement between tasks with the purpose of accomplishing that the execution of separate tasks is timely, in the right order and of the right quantity » (Reezigt, 1995 par Malone et crowston, 1994, p. 92).

[56] Un artefact est un outil artificiel conçu pour conserver, exposer et traiter l'information dans le but de satisfaire une fonction représentationnelle (Norman, 1993).

artifact (a permanent symbolic construct) in which the protocol is objectified.» (Schmidt et Simone, 1996, p 165)

Les récents travaux sur la coordination dans les systèmes de coopération font référence souvent référence aux définitions de Malone et Crowston (1994) qui considèrent la coordination comme le mécanisme qui consiste à gérer les interdépendances entre les activités :

> « *Coordination is the managing of dependencies between activities* » (Malone et Crowston, 1994, p. 92)

Les travaux de Malone et Crowston (1994) sur la coordination s'articulent autour de deux axes : la caractérisation des dépendances entre les activités et le choix des mécanismes de coordination en fonction du type de dépendance. Dans leur article « The Interdisciplinary Study of Coordination », les auteurs proposent une typologie des dépendances et des mécanismes de coordination (cf. Annexe 1)[57]. Cette typologie a par la suite été affinée dans des travaux plus récents des auteurs (cf. Malone et al., 1999 et Crowston, 2003). Ainsi Crowston (2003) distinguent six types de dépendances (cf. Annexe 2). Pour simplifier

- dépendance entre une tache et une ressource : une tache qui utilise un seul type de ressource ;
- dépendance entre une tache et de multiples ressources : une tache qui utilise de multiples ressources simultanément ;
- dépendance de partage : plusieurs tâches utilisent les mêmes ressources ;
- dépendance de flux : une tache utilise une ressource créée par une autre tâche ;
- dépendance d'ajustement : plusieurs taches créent la même ressource ;
- dépendance de composition entre tâche : décomposition d'une tâche en sous-tâches.

Pour simplifier l'interprétation cette typologie Crowston (2005) regroupe ces types de dépendances en deux catégories : dépendances entre une tâche et une ressource, dépendances entre deux tâches. Pour chaque catégorie il défini les mécanismes de coordination suivants :

- Gérer les dépendances entre une tache et une ressource (*a task-resource dependency*) consiste, à identifier les ressources nécessaires et disponibles, de

[57] Nous présentons la dernière typologie définie par l'un des deux auteurs (Crowston, 2003). La premire typologie sera présenté en annexe.

choisir les ressources appropriées et enfin affecter ces ressources aux acteurs chargées de réaliser ces tâches.

- Gérer les dépendances entre deux taches (*a task-task dependency*) nécessite que les tâches soient ordonnées en s'assurant que les sorties de la première tâche soient utilisées par la deuxième tâche.

A partir d'une analyse multidisciplinaire de la coordination (cf. tableau 2.2), nous pouvons dire que la coordination est l'objet de plusieurs définitions. Nous nous proposons dans les parties suivantes, de faire état des travaux issus de la littérature qui portent sur la coordination dans les communautés et plus spécifiquement dans les communautés Open Source.

Tableau 3.1 : Approche interdisciplinaire de la coordination

Auteurs	Définitions de la coordination	Disciplines
Chandler, 1962	*«structurer et faciliter les transaction entre des composantes interdépendantes».*	Théories des Organisation
Thompson, 1967	*«les protocoles, taches et mécanismes de prises de décisions désignés pour réaliser des actions entre des unités indépendantes».*	Théorie des Organisation
Mintzberg, 1982	*«la coordination est la colle de la structure, l'élément fondamental qui maintient l'ensemble les parties de l'organisation..elle s'exprime à partir de trois mécanismes : la supervision directe, l'ajustement mutuel et la standardisation ».*	Théorie des organisattion
Singh &Rein, 1992	*« la communication est un mécanisme d'intégration, d'harmonisation et d'ajustement de travail individuel vers un objectif vers un objectif large».*	CSCW
Reezigt,1995	«La communication consiste à *établir l'harmonisation entre les tâches dans le but d'assurer que ces tâches, séparées, soient réalisée en temps prévu, dans de bon ordre et correctement ».*	CSCW
Malone et Crowston, 1994; Crowston, 2003	*«la coordination a pour objectif de gérer les différentes dépendances entre activités et ressources».*	Théorie de la coordination
Schmidt et Simone, 1996	*« l'activité de coordination étant conçue comme une méta activité dontl'objectif est de gérer (c.-à-d. répartir, interfacer, etc.) les dépendances entre ressources, tâches, acteurs et activités d'un collectif».*	CSCW

I.B. La coordination dans les communautés

Les notions de communautés, soulignées dans le second chapitre, supposent des modèles d'action collective qui peuvent donner lieu à des problèmes de coordinations. Ces problèmes semblent être difficilement dépassés par le recours aux approches classiques des organisations[58]. Les communautés émergent donc avec de nouvelles modes de coordinations. Elle se fait, selon Muller (2004) par un couplage entre des normes sociales et l'existence de leaders communautaire. Pour Grosjean et Lacoste (1999), la coordination dans les communautés se base sur l'interaction et la communication entre ses membres.

I.B.1. La coordination par un couplage entre des normes sociales et l'existence de leaders communautaire

Dans le contexte des communautés, la coordination est souvent vue comme un ensemble de mécanismes pour établir des lien entre les acteurs, de maintenir les relations sociales entres eux et de gérer les rapports hiérarchiques (Grosjean et Lacoste, 1999).

Muller (2004) souligne à ce sujet que les normes sociales fournissent un premier mécanisme de coordination dans les communautés. Elles permettent selon lui de limiter l'accès aux individus en décrivant de manière générale les objectifs de la communauté ainsi que les moyens mis en œuvre pour les atteindre.

> « *les normes prévalant au sein d'une communauté permettent de délimiter un ensemble d'objectifs et de comportement considérés comme acceptables… la sélection des individus appelés à intégrer la communauté est similaire à un problème de correspondance entre les objectifs et comportements au sein de la*

[58] La théorie des coûts de transactions a permis de mettre la lumière sur les formes hiérarchiques de gouvernance comme mode de coordination répondant aux « échecs du marché ». L'existence de l'organisation est alors expliquée à travers l'existence de coûts de transaction (Coase, 1973 ; Williamson, 1975) que sont les coût résultant des ressources mobilisées pour accomplir une transaction par le biais du marché. L'analyse transactionnelle présente un ensemble de règles hiérarchiques qui prescrivent le comportement, guident le choix individuel conformément aux besoins de l'organisation et fournissent une structure commune. L'autorité et la hiérarchie constituent le mode régulateur permettant de coordonner l'activité et de réduire l'opportunisme (williamson, 1985).

communauté et les visées individuelles…Une des premières tâches entreprises par un nouveau membre consiste à assimiler les normes sociales au sein de la communauté à fin de mieux cerner les objectifs de la communauté ainsi que les moyens mis en œuvre pour les atteindre » Muller (2004, p. 54).

Les normes fournissent ainsi un mécanisme de coordination important en produisant une homogénéisation des comportements et des objectifs individuels. Elles constituent aussi, au sens de Arrow, une composante de l'autorité « impersonnelle » coordonnant les comportements individuels (Arrow, 1974). Ces normes sont partagées par les membres de la communauté et maintenues par l'existence de sanctions (morales) imposées aux individus la violant (Elster, 1995).

La coordination par les normes sociales est également valable dans le cadre des communautés Open Source. En effet, comme nous l'avions décrit dans la deuxième section de ce chapitre, l'intégration d'un nouveau membre dans la communauté des développeurs dépend du temps passé par l'individu à s'informer sur les normes sociales et les règles de comportement prévalant au sein de la communauté (von krogh et al., 2003).

D'un autre coté Muller (2004) souligne que la coordination par le biais des normes sociales donne lieu à certaines limites et qu'elle doit être couplée à la présence de leaders au sein de la communauté. Il montre que la présence de leadership dans une communauté peut jouer un rôle important dans la coordination des comportements d'agents hétérogènes appartenant à la communauté *« par une capacité de « médiation » qui leurs permet de filtrer les communications au sein des communautés à fin d'apporter une cohérence dans la base de connaissance commune et d'orienter le comportement des membres dans une direction déterminée* » (Muller, 2004, p. 57).

Seulement, à la différence des mécanismes de coordinations traditionnels ou les leaders sont désignés et assignés par une structure formelle, le processus de construction de leadership dans un cadre communautaire se base principalement sur deux mécanismes : la réputation et la confiance (Muller, 2004). Il s'agit de deux concepts complémentaires et étroitement liés permettant d'abord, par la réputation, la construction d'un statut de leader en facilitant les premières interactions entre les individus. Ensuite le statut de leader est soutenu par le développement de relations de confiances. « Ne fournissant qu'un mécanisme autorisant la première interaction, la réputation ne constitue qu'un mécanisme de court terme nécessaire à la construction de leadership. Un leader ne peut asseoir sa légitimité que dans le long terme par l'accumulation d'interactions avec les

membres de la communauté. C'est à ce point qu'agit la confiance. Cette dernière permettant la pérennisation des relations entre les leaders et les membres ».

Le concept de réputation dans les communautés est considéré comme un ensemble d'informations dans le comportement passé d'un individu. Selon Kreps (1990), la réputation est « un concept fortement utile dans des environnements distribuées caractérisés par de forte incertitude en la réduisant et en impliquant une première interaction entre agents » (Kreps, 1990).

Muller (2004) souligne également qu'en fournissant le point de départ des relations interpersonnelles, « *la réputation permet de à l'individu d'accroître sa notoriété au sein de la communauté. Dans le cas où ce dernier parvient à l'honorer, se forme un cercle vertueux où la multiplication des relations interpersonnelles contribue à un renforcement de la réputation, favorisant les interactions*. » (Muller, 2004, p. 65).

Dans une étude concernant les formes de coopérations interentreprises, Rullière et Torre (1996) montrent que la confiance est également un concept indispensable dans la coordination d'une activité collective. En effet, la répétition des échanges entre les acteurs favorise et maintien le comportement coopératif (Rullière et Torre 1996) par la construction d'antécédents communs qui donne naissance à la confiance provoquant à son tour les interactions futures.

Les travaux présentés dans cette partie montrent que la coordination dans les communautés se base sur les relations sociales et interpersonnelles qui facilitent les premières interactions, les maintient dans la durée. Ils donnent naissances à des dispositifs (confiance et réputation) qui permettent gérer les rapports hiérarchiques. Cependant, ces mécanismes diffèrent d'une communauté à une autre. Dans les communautés virtuelles, par exemple, la confiance qui émerge ne résulte pas de relations directes entre les acteurs. En plus, la construction de la réputation et de la confiance ne se fonde pas de manière exclusive sur le volume de connaissances diffusées mais aussi sur leurs qualités. Ceci est valable dans le cadre des communautés virtuelles, dans la mesure où le processus d'intégration ou de leadership dépend de plusieurs facteurs autre que la fréquence d'interactions (cf. section 2-chapitre II). Nous montrons dans la partie suivante que la communication, fondée sur la mise en œuvre des interactions entre les acteurs d'une communauté, est une fonction fondamentale de coordination.

I.B.2. La coordination par la communication

La communication est un phénomène général et présent dans toute communauté qui réuni des individus. « On ne peut pas ne pas communiquer » (Watzlawick et al. 1972). La communication est surtout l'élément indispensable à la coordination. Selon Leplat (2000), lorsque les individus se coordonnent, ils communiquent nécessairement soit en interagissant directement (interactions synchrones) ou en partageant des informations de manière différée (interactions asynchrones).

Dans les conceptions classiques (Shannon et Weaver, 1949), la communication est décrite comme la simple transmission d'information entre des interlocuteurs. La définition de Moles (1986) suggère que la communication résulte des éléments de connaissances que les individus ont en commun. L'action de communication dépend donc de la capacité des individus à faire correspondre leurs éléments de connaissances. Il définit ainsi la communication comme :

> « *l'action de faire participer un individu ou un système, situé en un point donné R, aux stimuli et aux expériences de l'environnement d'un autre individu ou système situé en un autre lieu et à une autre époque E, en utilisant les éléments de connaissances qu'ils ont en commun.*» (Moles, 1986, p. 25)

D'autres travaux sur la communication sont centrés sur les relations humaines qu'elle implique ou dont elle est à l'origine. Pour Muchielli (1981) et Cooley (1909), la communication est un processus qui permet aux individus de construire des liens entre eux, de maintenir les relations sociales entres les acteurs.

En ce qui nous concerne, nous nous intéressons aux communications fonctionnelles, c'est-à-dire aux communications qui concernent directement le contenu de la tâche réalisée (Grosjean et Lacoste, 1999). Elles permettent de faire « *circuler l'information, d'articuler l'activité, de planifier l'activité et de construire des référentiels commun* » (Trognon, 2004). Falzon (1994) étudie les communications fonctionnelles entant que dialogues entre deux ou plusieurs parties. Ces dialogues supposent l'élaboration d'un répertoire commun optimal. Dans cette élaboration il est nécessaire de prendre en compte de l'état du partenaire ou de l'interlocuteur. Selon Falzon (1994, p. 303), chaque partie doit d'une part, prélever le discours de l'autre des indices permettant de s'assurer du bon fonctionnement de la communication. Et d'autre part, fournir dans son discours des indices qui facilitent à son partenaire de réaliser la ou les tâches communes. Ces

indices sont sous formes d'accusé de réception, de demande de clarification, de reformulation, etc.

Les communications fonctionnelles occupent un rôle important dans l'activité de coordination des communautés. Puisqu'elles se produisent dans des situations de travail collectif qui nécessite que les actions soient partagées, articulées, modifiées, évaluée, etc. Selon Zarifian (1998) la communication intervient dans ces contextes (de travail coopératif) pour « *définir la nature des problèmes, identifier les objectifs, donner du sens aux actions, de faire converger les implications, et les motivations des individus* ». (Zarifian, 1998, p.115).

Karsenty (2000) a également montré le rôle de la communication dans les collectifs de travail. Plus spécifiquement, il défend l'idée que les besoins d'explication et les dialogues explicatifs, qui s'expriment à travers la communication, conduisent à enrichir la représentation partagée du problème en permettant la réinterprétation les modifications apportées à ce contexte. Il définit ainsi l'explication comme :

> « *un processus visant à modifier le contexte interprétatif, lequel est conçu comme l'ensemble de connaissances, croyances et attentes utilisées pour appréhender la situation de travail en cours. Les modifications apportées à ce contexte grâce à l'explication correspondent à une réinterprétation plus ou moins globale du problème à traiter.* » Karsenty (2000, p 289)

En ergonomie du travail, Savoyant et Leplat (1983) montrent également le rôle de la communication (le contenu du travail[59]) sur l'organisation du travail collectif. Ces auteurs définissent cinq fonctions de la communication dans la coordination d'une activité collective :

- *les communications d'orientation générale* : elles concernent le contenu des tâches et leurs règles d'exécution. Ce type de communication est généralement préalable à l'exécution effective de l'action ;

---

[59] L'analyse de contenu de travail est l'analyse qui porte sur l'activité. Elle détermine l'analyse de la tâche et les hypothèses sur l'activité. Une analyse de contenu ne peut être conduite indépendanement de l'analyse de la tâche et de l'analyse de l'activité. L'analyse de la tâche permet de définir les conditions nécessaures à l'activité mais ne peut pas prédire comment l'utilisateur va s'y prendre. Selon Leplat et Hoc (1983) la tâche désigne ce que l'individu doit faire : « l'ensemble des conditions objectives que le sujet est susceptible de prendre en compte dans la mise en jeu de sa conduite …elle concerne le le but à atteindre, les moyen disponibles pour y parvenir et les contrantes dans la mise en œuvre de ces moyens » (p.207) ».(Serge Agostinelli 1999, p25)

- *les communications de types* : elles permettent la verbalisation de certains éléments relatifs à l'activité du participant, dont l'objectif de fournir, au cours de la réalisation de l'action, des informations pour orienter la réalisation des tâches de manière coordonnée;
- *les communications de guidage* : elles définissent les tâches ou les éléments de l'activité d'un participant;
- *les communications de déclenchement des opérations* : elles ont pour objectif de localiser les moments d'exécution des tâches à réaliser ;
- *les communications de contrôle dans les réalisations de l'action collective* : elles vérifient au fur et à mesure la compatibilité des actions.

Selon Lacoste (1991), pour coordonner une activité collective, les communications s'organisent en trois types :

- *organisation en tours de parole* : elles déterminent l'alternance entre les opérations ou les acteurs ;
- *organisation séquentielle* : elles gèrent les échanges par les questions et les réponses. Ces formes de communications sont présentent en situations de dialogue et privilégiées dans le cadre de travail collectif (Falzon, 1994);
- *organisation de la réparation* : elles organisent le mode avec lequel les contenus sont corrigés, reformulé par les participants. Nous retrouvons ici l'idée développée par Karsenty (2000) selon laquelle l'explication joue un rôle dans la réinterprétation des réparations ou des modifications apportées par les participants.

Dans son étude sur l'usage des technologies d'information et communication éducatives (TICE), Agrostinelli (1999) montre que ces outils donnent naissances à de nouvelles formes de communications (les communications médiatées) qui contribuent à coordonner un travail en groupe tenu à distance. Il propose une définition de la *communication médiatée entre personnes appartenant à la même communauté* comme :

> « *Le processus qui ordonne la production, la diffusion et l'appropriation des informations, des connaissances au sein d'un espace collectif. Ces processus et procédures qui révèlent des échanges et symboles ou de significations entre les différents partenaires du réseau s'organisent autour d'une* communication *médiatée. Celle-ci fait intervenir un nombre indéfini de personnes dans des formes fixées pour tous ceux qui appartiennent à la même société ou à la même communauté…. La communication est nécessairement comprise ici comme*

l'articulation entre les deux niveaux de traitement de l'information celui de ma médiation humaine et cela de la médiation par l'objet technique.» (Agrostinelli, 1999, p. 25)

Cette définition de la communication médiatée est donc fondée sur la notion de collectif et sur des échanges humains supportés par un outil informatique. Agrostinelli (1999) distingue également plusieurs fonctions de la communication médiatée dans le déroulement et l'organisation d'une activité collective. En effet, l'auteur classe les messages émis par les interlocuteurs en catégories différenciées par leur fonctionnalité selon l'utilisation faite pendant le déroulement de l'action. Ainsi, il distingue :

- *les commentaires qui n'ont pas de lien direct avec l'activité* : ils transmettent juste un sentiment par rapport à la tâche. Il s'agit de situations d'échanges qui provoquent des interactions non finalisées dans la mesure où l'on ne font « *apparaitre aucune règle d'action ou procédure de résolution de problème* ».
- *les commentaires ayant pour but la verbalisation de l'activité* : ils informent les participants sur la réalisation d'une action. Leurs but c'est d'informer le partenaire de « *l'état d'avancement de l'exploration indépendamment de l'activité celui-ci* » ;
- *les messages sur le questionnement sur l'activité* : s'intéressent à l'activité sans énoncer clairement des éléments sur la résolution du problème (demande de contrôle par les pairs, demande d'information sur l'état d'avancement de l'exploration du partenaire, une impatience, un agacement par rapport au rythme des échanges) ;
- *les injonctions instrumentales :* présentent le cadre de la résolution (discours fondé sur la formulation et/ou la validation d'hypothèse, des procédures de résolution) ;
- *les confirmations ou les indications d'une solution partielle ou entière* : regroupent les messages dans lesquels la solution est explicitement formulée.

Les travaux présentés montrent que la communication dans les communautés et particulièrement dans les communautés méditées, ont une fonction importante dans la coordination de l'activité. Nous exposons dans la partie suivante les travaux qui étudient la coordination dans les communautés Open Source.

Section 2. La coordination au sein des communautés Open Source

La littérature distingue plusieurs formes de coordination sur lesquels les communautés Open Source reposent :

- un système de gouvernance reposant sur l'émergence de leaders ou de structures d'autorité et sur les processus de légitimation nécessaires à leur création et leur fonctionnement;
- l'usage systématique de la modularité;
- la communication médiatisée par l'usage des artefacts.

II.A. Légitimation du pouvoir et contrôle des contributions

Dans une étude sur la gouvernance des projets Open Source, Markus (2007) propose une définition selon laquelle les mécanismes de coordination dans l'Open Source sont perçus comme des solutions aux problèmes de contrôle et plus généralement des solutions pour gérer le développement du travail:

> « *In the operational coordination literature, OSS governance is understood as a solution to [… the problem of] loss of operational control and the solution is techniques for managing the process of OSS development work.* » (Markus, 2007, p. 156)

Comme nous l'avons souligné dans le second chapitre, la coordination dans les communautés Open Source implique une nature particulière de contrôle qui peut être centralisé au niveau d'un leader ou d'une équipe leader (Raymond, 1998 ; Lerner et Tirole, 2002) mais doit être constamment reproduite et légitimée par ceux qui constituent « les couches inférieures » de la pyramide. La légitimation n'est pas un concept statique, et doit être renouvelé à chaque fois (O'Mahony et Ferraro, 2007). D'après O'Mahony et Ferraro (2007), la structure de gouvernance des projets Open Source tel que Debain émerge progressivement et évolue, en fonction de la complexité du projet. La coordination de l'activité se base ainsi sur une structure d'autorité ou de contrôle qui émerge progressivement, mais encore une fois basée sur des mécanismes de légitimation.

Le deuxième trait caractéristique du développement Open Source est l'existence d'une division du travail. En effet, les différents contributeurs n'occupent pas les mêmes rôles

dans la communauté. Certains ont un rôle plus important que d'autres et influent par leurs décisions sur le devenir du projet. Ces contributeurs assurent la coordination du projet et contrôle son développement. L'étude de Mockus et al. (2000) sur le projet Apache montre qu'un noyau ou une équipe de petite taille (24 personnes) gère l'ensemble du projet. Nous avons également observé que l'intégration des contributions au code source du projet Mozilla n'est pas systématique mais nécessite un haut niveau de filtrage. Le filtrage est assuré par les responsables de modules qui veillent a ce que la version du code soie cohérente (cf. chapitreI-section 2). Moon et Sproull (2000) observent la même structure dans le projet Linux, puisque toute intégration de modification doit être validée par les responsables, et parfois même par le fondateur du projet Linus Thorvald lorsque les modifications concernent le noyau central du système d'exploitation.

Nous avons montré que le développement Open Source repose sur les contributions libres de participants dispersés géographiquement. Cependant, la coordination de l'activité nécessite un système de contrôle ou de filtrage de contributions assuré par les leaders ou les responsables des projets. Ces responsables ont pour fonction de décider quels traits devraient être intégrés dans le logiciel, quand et comment autoriser le changement du code source, et donner l'accès pour modifier le code (Raymond, 1999) :

> «*Core group membership can bestow some rights, including deciding what features should be integrated in the release of the software, when and how to empower other code maintainers, or to "pass the baton" to the next volunteer* ». (Raymond 1999, p 25)

Les responsables de projets disposent, selon Crowston et al. (2006), d'une autorité ou d'un pouvoir informel dont la légitimité n'est pas définie par une structure formelle mais par des mécanismes informels comme la réputation, les compétences et l'intensité des efforts. La fonction de l'équipe leader ressemble ainsi à celles d'une organisation traditionnelle. Néanmoins, le développement Open Source repose sur le principe de distribution du pouvoir (Crowston et al., 2006). En effet, ce principe donne la possibilité au contributeur externe (la périphérie) de devenir responsable par un mécanisme fondé sur les compétences (Crowston et al., 2006) :

> « *In comparison to traditional organizations, more people can share power and be involved in group activities*». (Crowston et al. ,2006, p. 567)

La distribution du pouvoir se fonde sur l'hypothèse selon laquelle la participation des utilisateurs (la périphérie) joue un rôle très important dans l'activité de développement

Open Source (Heckman et al., 2006). Selon Heckman et al. (2006), les utilisateurs contribuent au projet dans de multiples chemins, et deviennent une source cruciale de recrutement potentiel. Les intérêts des utilisateurs doivent, cependant correspondre aux objectifs du projet pour assurer son succès. La manière de gérer la relation entre les exigences de la périphérie et les valeurs des équipes représente donc un grand défi qui amène à s'interroger sur le degré de distribution du pouvoir entre la périphérie et l'équipe de développement.

D'après Crowston et al. (2006), un « pouvoir abusif » de la part des leaders, ou des décisions prises sur la base de préférences personnelles peut décourager les contributeurs (la périphérie) de s'investir plus dans la communauté:

> « *The first and most significant change firms make to FLOSS projects is to the authority structure by controlling decisions such as access to mailing lists, which code contributions are accepted and combined into the project and who should be empowered to be subproject leader or succeed the current leaders. As a result, volunteers may feel detached from open-source projects because they are not included in the decision making process.* » (Crowston et al., 2006)

Inversement, l'absence de leadership dans le cadre d'un développement distribuée rend la gestion de l'activité difficile. Lorsque les projets sont dirigés informellement, les contributeurs communiquent essentiellement de manière textuelle et les utilisateurs sont impliqués fortement dans l'activité du développement, le conflit est inévitable dans le cadre de développement logiciel. Le rôle des leaders est donc de résoudre les conflits, en assurant la cohésion du groupe et son efficacité.

II.B. Modularisation et persistance des interactions

La coordination des communautés Open Source est également réalisée par l'usage systématique de la modularisation. En décomposant le code en fragments (Torvalds, 1999), la modularisation permet de diviser le travail en plusieurs modules ou fragement. Les différents modules seront développés de façons indépendante et ensuite assemblés pour constituer le code source. La coordination du travail des développeurs opérant sur des modules différents n'est plus nécessaire puisque chaque module est développé séparément. Selon Torvalds (1999), le traitement du code en plusieurs modules permet de maîtriser la complexité de son développement et réduit le besoin de coordination. L'étude de Gosh et David (2003) montre que le projet Linux, comme la plupart des projets Open Source, repose sur une architecture composée de plusieurs

modules eux même développés par des équipes de petites tailles. Gosh et David (2003) observent que la majorité des développeurs travaillent sur un seul module. Ainsi, la modularisation permet la spécialisation des développeurs travaillant sur les mêmes modules.

En complément de la modularité des projets Open Source, la persistance des interactions permet aux contributeurs de pouvoir accéder aux décisions prises, aux négociations en cours et par suite de participer de manière appropriée au développement du projet. D'après Zacklad (2006), la persistance des interactions permet la construction de mémoire collective à partir de laquelle l'activité est organisée. Sous forme de document, la mémoire collective joue un double rôle: d'une part elle facilite la production des connaissances par la possibilité d'exploiter et de préserver les transactions communicationnelles distribuées. D'autre part, elle facilite l'organisation de l'activité générale puisque elle permet d'articuler entre l'ensemble des transactions produites (Zacklad, 2007)

D'un autre côté, la transparence des interactions permet l'identification du travail de chaque contributeur. Dans le développement libre, les lignes de codes ainsi que les contributions (par exemple les échanges de commentaires entre développeurs dans Bugzilla) sont signées par leurs auteurs ce qui permet d'identifier leurs différentes contributions et de juger de leurs qualités. Selon Féry (1997), l'indentification du travail permet également de détecter toute forme d'opportunisme des acteurs.

II.C. Communication médiatisée par les artefacts

L'infrastructure technique des projets Open Source assure une fonction de coordination. Elle n'est cependant pas suffisante. En effet, la modularisation favorise le travail solitaire et réduit le besoin de communiquer. Ainsi, chaque développeur travaille sur le module dont il est responsable sans avoir besoin de communiquer avec les autres développeurs ou responsables. Néanmoins, la modularisation pose problème lorsque des incompatibilités surviennent lors de la phase finale d'intégration. D'après Herbsleb et Grinter (1999), la modularisation dans les projets distribués n'élimine pas les besoins d'interactions et de communication. Ainsi, la coordination d'une activité distribuée, même modularisée, repose sur la communication. Malone et Crowston (1990) souligne que la coordination est une activité qui n'est pas directement observable. Elle est souvent étudiée à travers la communication en particulier dans des contextes où les

artefacts, par les e-mails, les forums ou des lignes de code, façonnent la structure d'interaction et favorisent le travail collectif. D'après David et Gosh (2008), la dépendance entre les différents modules du projet Linux implique des échanges importants entre les acteurs responsables de ces modules et que des liens sociaux peuvent se créer en raison de nombreuses collaborations. Dans une étude du projet Mozilla, Ripoche (2006) montre que les interactions participent à la réalisation du processus de coordination et que des variations dans ce processus sont reflétées par des variations dans les motifs d'interactions entre participants.

Dans la même veine, Gasser et Sandusky (2005) observent que la coordination des projets Open Source se fait à travers la négociation sur trois questions:

- « *Who's responsible for fixing this* ? »: La réponse à cette question permet de trier les bogues et les affecter à des composante spécifique. D'après Gasser et Sanduscky (2005), l'affectation du bogue nécessite une forte négociation entre les développeurs afin d'assurer l'efficacité de sont traitement.
- « *How should the BR be managed* ?»: Dans ce cas la résolution est gérer par une classification des bugs prioritaire dans la liste 'active bug reports'. Dans le cadre du développement Open Source, la classification des bugs se fait après négociations entre les membres de l'équipe de développement sur le niveau de priorité du bug.
- « *Is this a duplicate or dependent BR* ? »: Gasser et Sanduscky (2005) montrent également que la négociation entre les acteurs est une façon de gérer les dépendances entre les modules. La négociation entre les participants est ainsi déterminante pour le bon fonctionnement du logiciel. Avant d'intégrer des nouvelles lignes de code il faut que les responsables s'assurent que cette dernière n'affecte pas les autres modules. D'autre part, signaler un bug comme duplicate facilite la résolution en permettant l'utilisation de version déjà existante pour corriger le bug.

Selon Gasser et Sandusky (2005), d'une manière générale, la négociation joue un rôle important dans l'articulation du développement Open Source. Ainsi, les responsables de projets négocient pour trier les bogues, les affecter aux développeurs ayant les compétences nécessaires et valider enfin l'intégration des contributions qui garantissent la qualité finale du logiciel.

D'une manière générale, la coordination des projets Open Source, nécessite, à différentes situations, la négociation et l'interaction entre responsables à fin d'organiser

l'information et d'articuler le travail à l'appui du plus grand but de diriger le développement du projet.

Section 3. Synthèse et Apport à notre questionnement

Dans ce chapitre nous avons abordé la coordination comme un niveau fondamental de l'organisation d'une activité. L'étude de ce concept mené dans le premier chapitre, nous a permis de montrer en mobilisant différentes disciplines, que la coordination renvoie à la notion de gestion, de synchronisation et d'interconnexion, selon les théories des organisations (Chandler, 1962 ; Thompson, 1967; Mintzberg, 2000). Les recherches sur les systèmes de travail coopératif distribués (Schmidt et Simone, 1996) quand a eux ont porté sur les objets de la coordination. Ils ont permis d'identifier différents protocoles de coordination appuyée par ces objets. Les travaux de Malone et Crowston (1994) sur la coordination présentent l'avantage de mettre en évidence la manière de gérer les dépendances entre les tâches et les activités qui sont au cœur du problème de coordination. Ils proposent ainsi une typologie qui défini selon différents types de dépendances (tâche/activité, activité/activité, tâche/tâche), un choix des mécanismes de coordination à mettre en place.

Nous nous sommes également intéressés aux travaux qui étudient la coordination dans les communautés. Nous avons pu noter que la coordination dans ces contextes se fait par le biais des normes sociales, couplé par la présence de leadership (Muller, 2004). La coordination dans ces contextes et particulièrement dans les contextes médiatisés se fonde également sur la communication entre les acteurs (Leplat, 2000).

Dans le cas des communautés Open Source, la littérature présentée dans la seconde section, montre que la coordination ne peut pas s'effectuer par un contrôle hiérarchique mais sur la base de plusieurs formes de coordination qui doivent coexister pour assurer le déroulement de l'activité collective (Schmidt et Simone, 2000).

En effet, si de nombreux travaux montrent que la coordination dans le cadre du développement Open Source se fait par la formalisation de certains aspects du processus de coordination à l'aide de la modularisation (Gosh et David, 2003), ou grâce à un système de contrôle des contributions (Markus, 2007), ces formes de coordination peuvent être contournées (Schmidt, 1994) et doivent reposer sur la communication à travers l'interaction (David et Gosh, 2008) et la négociation entre les contributeurs (Gasser et Sandusky, 2005).

D’une manière générale, plusieurs travaux souligne le rôle fondamental de la communication dans l’organisation du travail collectif (Lacoste ,1991 ; Karsenty, 2000) et distinguent un nombre de ses fonctions notamment dans la coordination d’une activité collective (Savoyant et Leplat, 1983; Agrostinelli, 1999). Néanmoins, ces aspects n’ont pas été beaucoup étudiés dans le contexte du développement Open Source.

En effet, comme nous pouvons le constaté, malgré le nombre important des travaux portant sur l’organisation des projets Open Source, très peu sont les travaux qui adoptent une approche centrée sur la communication entre les acteurs pour étudier cette question. ***Une question fondamentale consiste alors à déterminer les fonctions de la communication dans l’organisation et la coordination de l’activité du développement.***

CHAPITRE IV- FONDEMENTS EPISTEMOLOGIQUES ET CHOIX METHODOLOGIQUES

TABLE DES MATIERES

CHAPITRE IV- FONDEMENTS EPISTEMOLOGIQUES ET CHOIX METHODOLOGIQUES

Tout travail de recherche repose sur une certaine vision, utilise une méthode, propose des résultats visant à prédire, prescrire, comprendre, construire ou expliquer un phénomène. La validité et la légitimité de ces résultats impose obligatoirement une réflexion épistémologique préalable et un choix méthodologique permettant de vérifier les données expérimentales (Wacheux, 1996 ; Thiétart et al., 1999).

L'objet de ce chapitre est de présenter les aspects épistémologiques et méthodologiques de notre travail. Ainsi, nous expliquons dans un premiers temps notre positionnement épistémologique de la recherche. Nous détaillons dans un second temps l'objet de notre recherche. Enfin, nous décrivons, dans un derniers temps la méthodologie adoptée pour étudier notre objet.

Section 1. Positionnement épistémologique

L'épistémologie s'interroge sur la nature, la méthode et la valeur de la connaissance à produire. Elle permet au chercheur de définir si la connaissance produite est objective, un reflet d'une réalité existante, ou une construction de la réalité.

En science de gestion, et plus généralement en sciences sociales nous pouvons identifier trois paradigmes : le positivisme, l'interprétativisme, et le constructivisme. La pluralité de paradigme conduit le chercheur à faire un choix. Il peut donc opter pour un choix de l'un des paradigmes et de s'y tenir. Dans ce cas le chercheur rejoint le point de vue des partisans de l'isolationnisme, selon lequel « *les différents paradigmes présents en théorie des organisations sont incommensurables et ne peuvent pas être réconciliés. Une conversation entre ces paradigmes n'est pas possible et ne devrait pas pouvoir être tentée.* » (Thiétart et al., 2003).

Le chercheur peut aussi tenter la réconciliation entre les différents paradigmes et la recherche d'un standard commun. C'est dans cet esprit que Miles et Huberman (1991),

appellent une position épistémologique aménagée, en donnant l'exemple d'une conception positiviste aménagée.

Enfin, pour une meilleure compréhension des phénomènes sociaux, le chercheur peut entreprendre une posture multi paradigme. Le chercheur bénéficie d'une diversité de paradigme pour construire et étudier son objet *« il dispose d'une variété d'approches qui, chacune à leur manière, sont en mesure de rendre compte de certains aspects des réalités complexes auxquelles s'intéressent les sciences de l'organisation* » (Koenig, 1993, p. 4).

Avant de définir notre positionnement épistémologique, nous présentons dans ce qui suit les paradigmes les plus utilisés en sciences de gestion, à savoir : le positivisme et le constructivisme.

I.A. Positivisme et constructivisme : deux conceptions différente

Le positivisme et le constructivisme représentent deux conceptions qui diffèrent largement sur la nature de la connaissance produite et le chemin de la connaissance emprunté. Concernant les positivistes, la réalité est considérée comme existante. Le travail du chercheur consiste alors à mesurer la réalité par l'observation de faits.

Au sens de Thiétart et al. (2003), la connaissance produite par les positivistes est objective et acontextuelle, « *elle correspond à la mise à jours de lois, d'une réalité immuable, extérieure à l'individu et indépendante du contexte d'interactions des acteurs* ». Contrairement au positivisme, dans le constructivisme, la réalité (sociale) est construite par les chercheurs grâce aux interactions entre acteurs, dans des contextes toujours particuliers (Berger et Luckman, 1996). En ce sens, dans le cadre du positivisme, le chercheur va découvrir des lois qui s'imposent aux acteurs. Dans le cadre du constructivisme, il va contribuer à construire, avec les acteurs, la réalité sociale.

I.B. Une posture épistémologique aménagée pour notre recherche

Comme nous l'avons vu dans la littérature, il n'existe pas encore de corps de recherche stabilité sur les questions relatives à la coordination des projets Open Source sous l'angle de la communication entre les acteurs. Nous adoptons ainsi une attitude d'observation et de description de situations de coordination dans le cadre du projet Mozilla.

De ce fait, notre étude ne suit pas un unique paradigme étant donné que nous cherchons à explorer et décrire un domaine de recherche particulier (vision constructiviste) et qu'au

même temps notre construction n'est pas indépendante de la réalité (vision positiviste). Nous allons ainsi nous situer dans une posture empruntant les deux logiques.

En effet, nous allons emprunter un chemin méthodologique semblable a celui développé par Miles et Huberman (1991), selon lesquels « *les phénomènes sociaux existent non seulement dans les esprits mais aussi dans le monde réel.*» (Miles et Huberman, 1991)

Cette position aménagée va nous permettre de décrire des formes et des faits de comportements sociaux grâce à une compréhension du jeu d'interactions des acteurs. Pour comprendre la réalité, il faut aussi une forte compréhension du contexte de l'interaction (Giordano, 1993). La connaissance ainsi produite sera subjective et contextuelle (Koenig, 1993). Dans cette démarche nous allons donc chercher à construire une réalité sociale à travers la compréhension des interactions entre des acteurs dans un contexte spécifique qui est la résolution de problèmes.

I.C. Orientation de la Recherche : une stratégie hybride de découverte

Après avoir réfléchi sur les questions relatives au positionnement épistémologique, pour poursuivre cette démarche méthodologique, nous devons répondre à la question : *Comment nous allons construire la connaissance ?*

II.C.1. La dichotomie exploration et test : adopter une démarche inductive, abductive ou déductive

Selon Charreire et Durieux (2003), le processus de construction de la connaissance peut s'opérer par deux voies : l'exploration et le test. Les auteurs soulignent qu' « *explorer en management consiste à découvrir ou approfondir une structure ou un fonctionnement pour servir de grands objectifs : la recherche de l'explication (et de la prédiction) et la recherche d'une compréhension. Explorer répond à l'intention initiale du chercheur de proposer des résultats théoriques novateurs, c'est-à-dire de créer de nouvelles articulations théoriques entre des concepts et/ou d'intégrer de nouveaux concepts dans un champ théorique donné* » (Charreire et Durieux, 2003).

D'après Charreire et Durieux (2000), le test permet au chercheur de valider des hypothèses par la confrontation à la réalité. Le travail du chercheur consiste alors à expliquer la pertinence de sont hypothèse :

« *Tester est l'ensemble des opérations par lesquelles le chercheur met à l'épreuve de la réalité un ou des objets théoriques ou méthodologiques. L'objectif est de produire une explication par l'évaluation de la pertinence d'une hypothèse, d'un modèle ou d'une théorie dans un but d'explication*. » (Charreire et Durieux, 2000).

L'orientation vers l'un des deux processus de construction de la connaissance (le test ou l'exploration) n'est pas indépendante du positionnement épistémologique. En effet, Charreire et Durieux (1999) souligne que « *l'exploration se réfère à une démarche inductive et/ou abductive alors que le test fait appel à une démarche de type déductive* ». (Charreire et Durieux,1999)

L'induction se fonde sur un raisonnement par lequel on passe du particulier au général. On parle alors d'induction si, à partir d'une vérification de relation sur un certain nombre d'exemples concrets, nous pouvons dire que la relation est valide pour toutes les observations.

A l'inverse, la déduction s'appuie sur une démarche qui consiste à élaborer une ou plusieurs hypothèses et à les confronter ensuite à une réalité. Le but est alors de porter un jugement sur la pertinence de l'hypothèse initialement formulée (Charreire et Durieux, 2000).

Suivre une démarche abductive quand à elle n'a pas pour objectif de tester des théories ou des modèles mais de produire de nouvelles conceptualisations théoriques. Elle permet d'explorer un objet par un aller-entre l'observation de la réalité et les connaissances théoriques, tout au long de la recherche (Koenig, 1993). D'après Koenig (1993), cette démarche est pertinente lorsqu'on souhaite explorer un contexte complexe et peu explorer.

I.C.2. L'abduction comme démarche adoptée pour notre recherche

Notre démarche de recherche s'est basée sur une démarche abductive consistant à procéder par allers-retours entre le terrain et la littérature scientifique. Ce choix s'explique par deux raisons :

D'abord, la disponibilité des données et la facilité d'y accéder nous ont poussé à débuter notre analyse par l'exploration du terrain. En effet, nos analyses ont porté sur une sélection de rapports de bogues issues de Bugzilla, le système de suivie de bogues du projet Mozilla. Comme expliqué, chaque rapport retrace les échanges, interactions et négociations entre les contributeurs en rapport avec le traitement du bogue.

Ensuite, comme nous l'avons révélé dans les chapitres précédents, la littérature scientifique relatives à l'organisation des communautés Open Source est très riche. Elle représente, de ce fait, un cadre de référence sur lequel nous nous sommes appuyé pour identifier les problématiques les plus importantes.

Ainsi, notre recherche satisfait les deux critères d'une approche abductive, puisqu'elle se construit sur la base des interprétations issues de la réalité, à savoir les interactions entre les contributeurs dans des contextes de résolution de problèmes. Au même temps cette recherche s'appuie sur des concepts théoriques pluridisciplinaires.

Notre démarche a commencé par une première incursion sur le terrain. Notre **objectif été d'identifier les fonctions de la communication dans l'organisation de la résolution de bogues au sein de Bugzilla.**

Cette incursion s'est basée sur une analyse exploratoire d'un exemple de rapport de bogue identifié sous le numéro : BR-362919. L'étude de ce rapport nous a permis de nous familiariser avec notre terrain de recherche. En outre, il a permis l'identification de certaines propriétés de l'organisation de la résolution de problèmes au sein de Bugzilla.

Néanmoins, pour justifier nos observations il ne suffit pas d'étudier un seul rapport de bogues. Ainsi, nous avons effectué une analyse linguistique sur une sélection de 694 rapports de bogues. Cette analyse a conduit, d'une part, à préciser les observations faites suites à notre première étude inductive du rapport BR-362919, et d'autre part, à développer les questions de recherche que nous souhaitons étudier.

L'analyse linguistique des 694 rapports de bogues confrontés à notre première étude inductive nous ont amené à identifier quartes fonctions de la communication dans l'organisation de résolution de bogues au sein de Bugzilla : ***la spécification des tâches, la définition des rôles, l'explicitation de l'activité, et l'articulation de l'activité.*** En effet, ces fonctions ont été identifiées à travers une analyse du langage utilisé par les participants dans leurs échanges. Comme nous le verrons plu tard, c'est les différentes propriétés du langage utilisé par les contributeurs qui nous ont permis d'identifier les fonctions de la communication au sein de Bugzilla.

Pour orienter notre recherche, nous avons décidé d'effectuer une revue de la littérature sur l'organisation des communautés Open Source. Notre but été d'identifier les concepts qui permettent de donner sens à nos observations et de nous diriger vers de nouvelles interrogations.

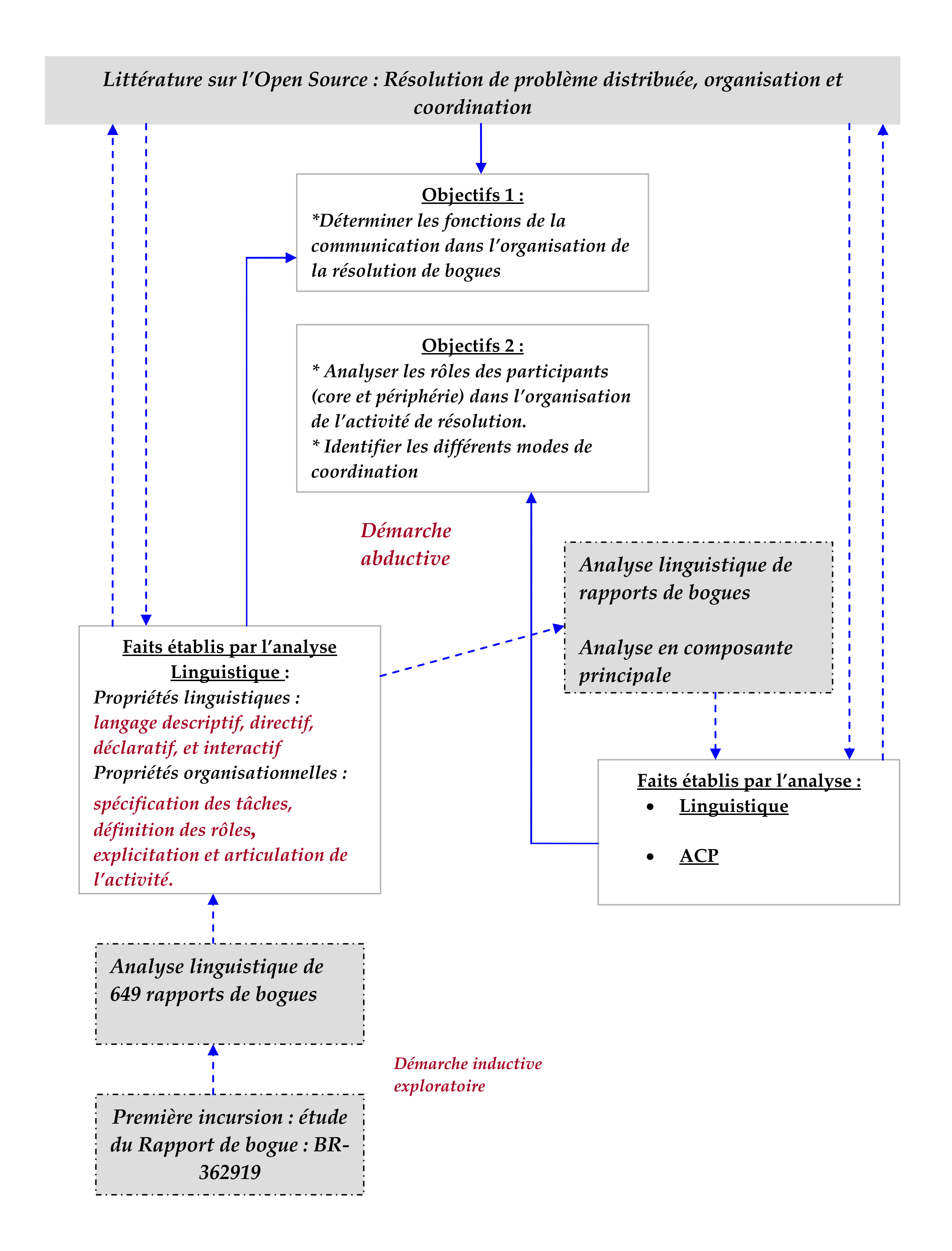
A l'issue de cette revue un nouveau concept a été introduit dans notre analyse : le statut du contributeurs dans la communauté. Nous avons ainsi rejoint les travaux qui mettent en exergue le fait que les contributeurs ne possèdent pas le même statut dans la communauté (Mockus et Herbsleb, 2002; Von Krogh et al., 2003, Crowston et Howison, 2005). Ce concept issu de la littérature scientifique a été présenté dans le second chapitre et fera l'objet d'une présentation détaillée dans le cinquième chapitre (section 1).

Dans le cadre de notre travail nous avons choisit de distinguer entre deux statuts : les contributeurs du **cœur** et les contributeur de la **périphérie**. En se basant sur cette distinction, notre recherche s'est orientée vers les propositions suivantes :

- ***Les caractéristiques du langage utilisé dans Bugzilla varient en fonction de l'identité de son auteur dans la communauté (cœur ou périphérie).***
- ***La variation des caractéristiques linguistiques identifie le rôle des participants dans l'organisation de l'activité au sein de Bugzilla.***
- ***La variation des caractéristiques linguistiques implique différents modes de coordinations.***

L'émergence des propositions de recherches s'est donc basée sur une démarche d'abduction. Nous suivons ainsi le raisonnement de Charreire et Durieux (2003, p. 63) schématisé dans le figure 4.2 ci-dessous.

Figure 4.2 : Démarche de recherche adoptée



Au terme de cette première démarche nous nous sommes donc intéressés à l'organisation de l'activité de résolution de problème au sein de Bugzilla, un nombre de questions de recherche ont émergé. Le tableau 4.2 suivant présente les objectifs, les propositions et les questions de recherche.

Tableau 4.1 : Objectifs, propositions et questions de recherche

Objectifs	Propositions	Questions de recherche
Déterminer les fonctions de la communication dans l'organisation de la résolution de bogues.	*La communication par le langage occupe une place centrale dans l'organisation de la résolution de bogues.*	***Q1 : Quelles sont les caractéristiques du langage utilisé par les contributeurs dans Bugzilla ?*** ***Q2 : Quelles sont les fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla ?***
Analyse les rôles des participants (core et périphérie) dans l'organisation de l'activité de résolution.	*Les caractéristiques du langage utilisé par les participants du cœur et de la périphérie sont différentes, cette différence peut nous aider à identifier leurs rôles dans l'organisation de l'activité.*	***Q3 : Quelles sont les propriétés du langage utilisé par les participants du « cœur » et de la « périphérie » ?*** ***Q4 : Quelles sont les rôle des participants du « cœur » et de la « périphérie » dans l'organisation de l'activité au sein de Bugzilla ?***
Identification des différents modes de coordination.	*la variation du langage peut impliquer différents modes de coordinations.*	***Q5 : Est-ce que la variation du langage utilisé dans Bugzilla implique des modes de coordination différents ?***

Section 2. L'objet de notre recherche

D'après Allard Poesi et Maréchal (2003), l'objet de recherche est la suite de l'existence d'un problème spécifique « *c'est à travers ce dernier que le chercheur interroge les aspects de la réalité qu'il souhaite découvrir, ou qu'il tente de développer une compréhension de la réalité.* » (Allard Poesi et Maréchal, 2003, p.41). Ils expliquent également que l'objet de recherche détermine le type de la contribution d'un travail de recherche :

« L'objet de recherche exprime indirectement le type de contribution que la recherche va offrir : contribution plutôt théorique, méthodologique ou empirique.» (Allard Poesi et Maréchal, 2003, p.41)

L'objet de notre recherche est **d'étudier l'organisation des projets Open Source en rapport avec la résolution de bogues par l'intermédiaire d'une approche centrée sur la communication entre les acteurs.** Dans ce cadre, nous avons choisi d'étudier le projet Mozilla. Ceci à travers l'observation des échanges entre les contributeurs dans 694 rapports de bogues identifiés a partir du système de suivi de bogues Bugzilla.

Comme expliqué notre démarche de recherche est abductive, nos interrogations émergent ainsi d'une part du terrain, c'est-à-dire d'une réalité observée, et d'autre part de la littérature théorique dans le but de mieux comprendre et d'expliquer nos observations empiriques. Nous avons ainsi pu dégager des propositions en se basant sur les résultats de notre travail empirique appuyé par des connaissances issues de cadres théoriques. Nous pouvons donc dire que notre contribution est plus empirique.

II.A. Intérêts d'étudier le projet Mozilla/Bugzilla

Notre étude porte sur la communauté Mozilla. Plus précisément, nous allons analyser les rapports de bogues retraçant les échanges entre les contributeurs. Plusieurs raisons expliquent notre choix :

- D'abord, le succès du projet Mozilla implique des interrogations sur son mode d'organisation, sa structure de participation, et ses pratiques spécifiques, qui le distingue des projets traditionnels ;
- Ensuite, malgré le nombre des travaux portant sur les projets Open Source en générale et la communauté Mozilla en particulier, Il existe peu de travaux qui les étudient sous les différents aspects que nous avons définis ;
- Enfin, la disponibilité des données sur le site www. Bugzilla. Mozilla .com, nous permet de récupérer les données librement. En plus la persistance des traces dans Bugzilla nous permet de constituer les traces des échanges entre acteurs.

II.B. Rappel sur le fonctionnement du projet Mozilla

Figure 4.3 : Vue d'ensemble sur le projet Mozilla

Decision-Makers
Firefox
Module Owners
Thunderbird
GOUVERNANCE AND ROLES
Super-Reviewers
PRODUCTS
Release Drivers
mozilla
Component Owners
Bug reports –bug activity
BUGZILLA
Current Developers
CODE
ROLES
Bug reportor
Quality Assurance
Bug assignee
Developer Network
Repositories
Contributors
Leaders

Section 3. Choix méthodologiques : Un plan d'étude multi-méthode

Pour étudier notre projet nous avons choisi un plan d'analyse qui se base sur les deux démarches : **quantitative et qualitative (**cf. figure 4.1**)**. En effet, pour répondre aux questions de recherches, nous avons choisit dans un premier temps, une **démarche qualitative exploratoire**. Ce choix s'explique par la volonté d'avoir une vue sur la réalité des échanges entre les participants, d'essayer d'apporter des réponses aux questionnements tirées de la littérature.

Dans une seconde partie une **démarche quantitative** s'avère nécessaire. L'objectif de cette démarche est double : nous tentons, d'une part de développer et d'approfondir notre étude sur la communication entre les contributeurs, d'autre part de valider nos observations tirées de l'analyse qualitative.

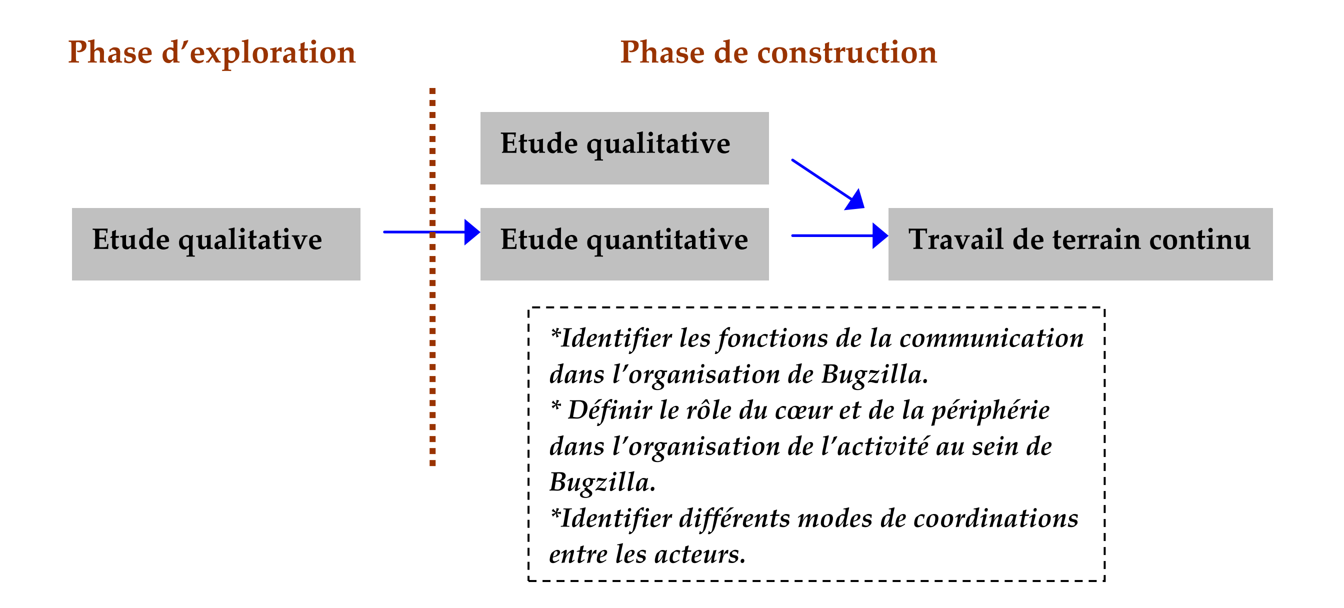
Puisque notre analyse porte sur la communication entre les contributeurs, la nature des données mobilisées est donc textuelle. Pour quantifier ces données qualitatives, il existe plusieurs méthodes bien développées de traitement quantitatif des données qualitatives textuelles. Nous avons choisit la méthode de l'analyse linguistique basée sur le corpus.

En effet, l'utilisation de cette méthode nous a permis, dans une première étape, par le biais d'outils informatiques, de traiter statistiquement nos données (textuelles). Nous avons ainsi proposé une représentation chiffrée des caractéristiques linguistiques du texte étudiée.

Cette représentation a elle-même fait l'objet, dans une seconde étape, d'une interprétation qualitative. Notre but été de définir les propriétés du langage utilisé au sein de Bugzilla, et de déterminer par conséquent les fonctions de la communication dans l'organisation de l'activité.

Nous détaillons dans les chapitres suivant cette méthode ainsi que les différentes étapes de recueil et de traitement de nos données.

Figure 4.4 : Démarche utilisée pour étudier l'organisation de Mozilla



Section 4. Conclusion et synthèse sur les différentes étapes de la recherche

Comme le synthétise le tableau 4.2 décrivant le design de la recherche, notre étude a débuté par une première incursion sur le terrain de recherche. Cette incursion a fait l'objet d'une analyse qualitative. Notre objectif été d'***identifier les fonctions de la communication dans l'organisation de la résolution de problème au sein de Bugzilla***. Quatre fonctions ont émergé du terrain : ***spécifier les tâches, définir les rôles, expliciter l'activité,*** et ***articuler l'activité***.

Les résultats de l'analyse qualitative, appuyé par une revue de la littérature scientifique sur les questions relatives à l'organisation des communautés Open Source ont orienté par la suite une démarche de recherche quantitative. L'analyse quantitative s'est intéressée aux questions suivantes :

- ***Les caractéristiques du langage utilisé dans Bugzilla varient en fonction de l'identité de son auteur dans la communauté (cœur ou périphérie).***

- ***La variation des caractéristiques linguistiques identifie le rôle des participants dans l'organisation de l'activité au sein de Bugzilla.***
- ***La variation des caractéristiques linguistiques implique différents modes de coordinations.***

Ces précisions épistémologiques et méthodologiques étant apportées, nous présentons dans les chapitres suivant les outils et méthodes utilisées pour recueillir et traiter nos données ainsi que les résultats de nos analyses.

Tableau 4.2 : Architecture de la recherche

Echéances	2006	2007	2008	2009	2010	2011
travail théorique	**Première revue de la littérature** **Sur les communautés Open Source** **Cadre général de l'étude:** *** La résolution de problèmes distribuée**		**Retour à la littérature** **Nouveaux concepts :** **** Le statut des contributeurs*** **** la communication comme fonction de coordination***			**Rédaction définitive**
		Construction de l'objet d'étude				
travail empirique	**Le choix de travailler sur le projet Mozilla (le collectif Bugzilla)**	**Analyse qualitative exploratoire**		**Analyse linguistique quantitative**	**Analyse en composante principale**	
						Tentative de modélisation empirique (voir annexe)
Résultats		**Les fonctions de la communication dans l'organisation de l'activité :** **Spécifier les tâches* **Définir les rôles* ** Expliciter l'activité* ** Articuler l'activité.*		**Le rôle des participants dans l'organisation de l'activité :** **Le rôle du cœur* *Énoncer l'action* *Décrire le contexte* *Diriger l'action* *Légitimer l'action* **Le rôle de la périphérie* *Décrire le contexte* *recontextualiser les interactions*	**Trois modes de coordinations entre participants du cœur et de la périphérie :** ** mode cœur vs périphérie* **mode forum vs cvs* **mode interactif vs solitaire*	

DEUXIEME PARTIE:
PRESENTATIONS DES RESULTATS EMPIRIQUES

DEUXIEME PARTIE : PRESENTATION DES RESULTATS EMPIRIQUES

CHAPITRE V-ANALYSE DU LANGAGE POUR IDENTIFIER LES FONCTIONS DE LA COMMUNICATION DANS L'ORGANISATION DE L'ACTIVITE AU SEIN DE BUGZILLA

Section 1. *Approche et Procédure d'analyse*
Section 2. *Analyses et résultats*
Section 3. *Conclusions et affinement de la problématique*

CHAPITRE VI-L'ORGANISATION DE LA RESOLUTION DE PROBLEME DANS BUGZILLA : ROLES ET MODES DE COORDINATION

Section1. Procédure de l'analyse
Section 2. Résultats de recherches : *analyses et conclusions*
Section 3. Conclusions

CHAPITRE V- ANALYSE DU LANGAGE POUR IDENTIFIER LES FONCTIONS DE LA COMMUNICATION DANS L'ORGANISATION DE L'ACTIVITE AU SEIN DE BUGZILLA

TABLE DES MATIERES

CHAPITRE V- ANALYSE DU LANGAGE POUR IDENTIFIER LES FONCTIONS DE LA COMMUNICATION DANS L'ORGANISATION DE L' ACTIVITE AU SEIN DE BUGZILLA

« *Processes of producing and understanding discourse are matters of human feeling and human interaction. And understanding of these processes in language will contribute to a rational as well as ethical and humane basis for understanding what it means to be human* »
(Biber, 2009)

Après avoir mis en évidence dans le chapitre IV, les questions de recherches qui sont au cœur de cette thèse y sont associées ainsi que les aspects pratiques de notre travail, nous présentons dans le présent chapitre les résultats de notre étude exploratoire de Bugzilla.
Les recherches précédentes identifient les rapports de bogues comme la voie principale par laquelle la communauté de développeurs communiquent, collaborent et coordonnent l'activité (Mockus et al., 2002). En supposant que l'organisation d'une communauté, est reflétée par les outils avec lesquels elle coordonne son activité (Lanzara et Morner, 2005), les aspects organisationnels de cette communauté doivent être visible à travers ces outils de communication ; ou plus exactement dans le langage produit dans ce contexte.

Cette partie de notre travail, vise à étudier les caractéristiques du langage utilisé par les contributeurs au sein de Bugzilla, ainsi nous nous intéressons au langage en tant qu'ensemble. Plus spécifiquement, nous tentons d'identifier les fonctions de la communication dans l'organisation de l'activité.

Nous étudions le langage en abordant les questions de recherches suivantes:

Q1 : Quelles sont les caractéristiques du langage utilisé par les contributeurs dans Bugzilla ?

Q2 : Quelles sont les fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla ?

Dans une première section de ce chapitre nous exposons le cadre de l'analyse ainsi que des procédures mises en œuvre. Après une description des principales approches utilisées pour étudier les communications médiatisées, nous finissons cette section en décrivant la procédure de travail. La seconde section est consacrée à la présentation des résultats de cette étude. L'interprétation des résultats fera l'objet de la même section.

Section 1 : Approche et Procédure d'analyse

Les communications médiatisées ont suscité l'attention d'un nombre important de chercheurs. Leurs objectifs a été de décrire et d'analyser les échanges qui se produisent dans divers contextes médiatisés tel que les forums de discussion, les messageries, etc. Plusieurs méthodes ont été développées pour étudier les communications médiatisées. Dans le cadre de notre travail, la communication étant exclusivement non verbale, les données générées sont donc textuelles.

Nous nous intéressons donc, aux méthodes qui analysent les données textuelles. D'après Fallery et Rodhain (2007), l'analyse de données textuelles désormais appelée (A.D.T) regroupent quatre approches *« visant à découvrir l'information « essentielle » contenue dans un texte* » (Fallery et Rodhain, 2007, p2). Ces méthodes se distinguent, selon Fallery et Rodhain, par le niveau d'analyse souhaité :

- « *De quoi parle-t-on ? C'est le domaine de l'analyse lexicale ;*
- *Comment en parle-t-on ? il s'agit de l'analyse linguistique ;*
- *Comment représenter une pensée ? C'est l'ambition de la cartographie cognitive ;*
- *Et enfin comment interpréter un contenu ? Il s'agit de l'assistance thématique.* » (Fallery et Rodhain, 2007, p3).

Les méthodes les plus utilisées, et les moins complexes sont les méthodes d'analyse lexicale ou lexicographiques. Ces méthodes analysent un langage en procédant par un comptage des mots, formes ou segments de formes d'un texte. Elles sont rendues faciles grâce au développement de l'informatique qui permet le traitement (comptage) et stockage d'une quantité importante de données textuelles.

Les méthodes les plus sophistiquées sont les méthodes d'analyses linguistiques. De la linguistique traditionnelle, à la linguistique qui mobilise les outils de traitements automatique des langues (TAL). Les analystes se sont dotés de divers outils d'analyses et de descriptions empiriques pour étudier les communications à travers l'analyse du langage. D'après Manning (2002), analyser un langage consiste à déterminer une 'structure cachée' dans le but de tester des hypothèses ou de mettre en évidence des phénomènes. Elle repose selon Fallery et Rodhain (2007), sur l'existence de relation entre un système linguistique et un système cognitif :

> « *Il s'agit de prendre en charge à la fois les aspects liés à la cohérence référentielle (ce à quoi le texte se réfère : des substantifs, signes linguistiques qui renvoient à une réalité extra linguistique) et aussi ceux relatifs au contexte d'énonciation (comment est-ce dit : des verbes, des adverbes, des conjonctions, des connecteurs..qui servent à traduire la relation du locuteur à LA situation, son point de vue et ses jugements).* » (Fallery et Rodhain, 2007, p11)

Dans ce chapitre, notre objectif est étudier la communication à travers l'observation de phénomènes linguistiques et ce, indépendamment de théories linguistiques précises. Nous adoptons ainsi une démarche inductive, visant à révéler les propriétés de la communication, dans le cadre de Bugzilla. Dans cette optique, la littérature distingue deux démarches pour étudier les communications médiatisées.

La première démarche, inductive, et dans laquelle nous nous inscrivons, est purement exploratoire. Elle consiste à identifier des catégories ou types de langages, à travers l'analyse de la variation linguistique.

Cette démarche concerne un langage diversifié; elle est rendue plus simple par les possibilités de traitement automatique des données textuelles. Ces possibilités résultent de la diffusion de l'informatique et des nouvelles technologies, qui permettent aux chercheurs, non seulement d'accéder à des ressources numérisées, mais leur donne également la possibilité de former des corpus regroupant un volume important de textes numérisés. Nous retrouvons dans ce domaine, et en particulier en linguistique anglophone, d'importants travaux dans lesquels l'informatique en tant qu'outil joue un rôle central. Ces travaux émergent d'un nouveau courant appelé « la linguistique sur corpus », où les chercheurs étudient les variations linguistiques dans un corpus de texte, ceci en mobilisant des outils de traitement statistique. Leur objectif est de déterminer une typologie de textes,

à travers l'étude des variations linguistiques entre différents textes hétérogènes, formant un corpus. Un corpus étant défini comme une « *collection de données du langage empirique, de textes (ou de fragments de textes), qui sont des échantillons d'un discours donné, dotés en conséquence d'une valeur représentative* ». (Wolfgang Teubert par Aurélie Lebaud, 2009).

Contrairement à la première, La deuxième démarche vise à définir un type de langage ou texte, par correspondance avec des typologies préexistantes. Selon la logique de Bronckart (1996), cette démarche consiste à faire des rapprochements entre un langage à étudier et des types de langage préexistants ou répertoriés.

Chaque type de langage correspond à une situation de communication distincte, et déterminé par la cooccurrence d'un certain nombre de traits linguistiques. Ces traits linguistiques peuvent correspondre à une classe lexicale (nom, verbe, adjectif, etc.), un genre (masculin, féminin, ect.), un nombre (singulier, pluriel, etc.), une personne (première, deuxième, troisième), un temps (impératif, présent, futur antérieur), etc.

Herring (2007) propose une classification des typologies de langage utilisées, pour étudier les communications. Comme l'indique le tableau 5.1, Herring (2007) distingue quatre critères de classifications : le mode de production, le déroulement de la communication, le type et le genre du texte. A chaque critère correspond une typologie de texte étudié dans la littérature.

Biber (1989) avant lui, distingue les textes « narratifs » et « expositifs ». De même, il ne considère pas un type unique d'interaction mais en distingue deux : l'interaction interpersonnelle intime et l'interaction informationnelle.

Bronckart (1996), pour sa part propose une typologie selon la situation d'énonciation, il définit par ailleurs trois types ou « architypes » de langage : le langage théorique, le langage en situation et la narration.

Tableau 5.1: Typologies Approches pour étudier les communications (Herring, 2007, p.5)

Critères de classifications	**Types**	**Auteurs**
Mode de production	Parlé, écrit	Chafe and Danielewicz (1987), Ferrera (1991)
Déroulement de la communication	Monologue, Asynchrone dialogue (Synchrone), polylogue	Dooley and Levinsohn (2001)
Type de texte/discours	Conversation, narrative, exhortation, exposition, etc.	Longacre (1996), Virtanen (1992)
Genre du texte /Registre	Chat, interview, lecture publique, lettre personelle, travail scientifique, article, etc.	Biber (1988, 2009)

Dans une première partie de cette section, nous présentons notre approche pour analyser nos données textuelles : ***une approche linguistique s'inscrivant dans une démarche purement inductive***. De manière spécifique, nous adhérons au courant de la « linguistique sur corpus » en s'attachant particulièrement à l'approche largement empirique de Biber (1988, 2009). Nous justifions dans une seconde partie, le choix de cette approche, et présentons la procédure d'analyse.

I.A. La linguistique sur corpus: études sur l'usage du langage par l'identification de genres ou registres linguistiques

Biber et al. (1998) classent les études sur le langage en deux grands domaines : les études sur la structure du langage « *studies of structure* » et les études sur l'usage du langage « *studies of use* ».

Traditionnellement, les analyses linguistiques traitent la structure du langage en identifiant les unités qui déterminent la structure d'un langage (mots, phrases, classes grammaticales, etc.). Ces analyses décrivent comment ces unités se combinent pour former de plus grandes unités grammaticales (par exemple, comment les mots peuvent être combinés pour former des phrases, les phrases pour former des clauses, etc.).

Une perspective différente, consiste à analyser le langage en termes d'utilisation «*langage use*». Dans cette perspective, l'intérêt est d'identifier, dans un contexte déterminé, un style du langage utilisé par un auteur « *individual author's style* » ou dans une variété de textes (textes produits dans des situations différentes). Les études de Koppel et Argomon, (2002) et Herring et Paolillo (2006) portent par exemple leurs comparaisons, sur le

langage utilisé par les hommes et les femmes dans les forums électroniques, en identifiant différents styles dans l'utilisation du langage. D'autres études comparent le langage utilisé dans différents textes : chaque texte est produit dans une situation différente (une lettre adressée à un ami, un article écrit dans un journal, un article académique, etc.) et les variétés du langage utilisées dans ces différentes situations sont dénomées des registres. D'après Biber et al (1998), identifier les caractéristiques de ces registres, genres ou types linguistiques, reste complexe d'un point de vue méthodologique, en raisons du nombre de choix grammaticaux et lexicaux qui entrent en jeu, et des besoins de représentativité qui poussent les analystes à mobiliser une grande quantité de données textuelles.

> « *How can we find the patterns in the language used in conversation, newspapers, academic prose, personal letters, etc? How can we characterize the language used in these different varieties? Questions such as these are complex and an important aspect of studies of use…It is also a complex one because many different grammatical and lexical choices come into play.* » (Biber et al, 1998, p 3)

C'est en raison de ces difficultés que l'approche de la linguistique sur corpus a été développée.

Nous présentons dans ce qui suit les principes de cette approche, puis nous résumons par la suite les travaux qui étudient l'usage du langage dans le contexte des forums électroniques.

I .A.1 Principes

I .A .1.a Une approche basée sur le corpus

Les principales caractéristiques d'une approche basée sur le corpus sont les suivantes :

- elle est empirique, c'est-à-dire qu'elle s'appuie sur l'analyse des tendances réelles de données textuelles ;
- elle se base sur un volume important de données textuelles formant un corpus ;
- elle utilise l'informatique comme outil dont l'objectif de traiter automatiquement ces données ;
- elle mêle à des techniques quantitatives et qualitatives.

L'avantage de cette approche, provient d'abord de la possibilité d'utiliser les outils informatiques qui permettent d'analyser et de stocker de grandes bases de données textuelles. Il est important de noter que l'analyse basée sur le corpus, doit aller au delà du

traitement statistique, qui donne une représentation chiffrée des traits linguistique. Elle inclut surtout une interprétation qualitative de ces représentations :

> « *The goal of corpus-based investigations is nor simply to report quantitative findings, but to explore the importance of these findings for learning about the patterns of language use.* » (Biber et al, 1998, p 4)

I.A .1.b Identification de registres par l'étude des variations linguistiques

Pour identifier un registre, Biber (1988) part de l'idée que les textes peuvent varier dans plusieurs de leurs traits linguistiques (lexicale, syntaxique) en fonction de la situation. Il défini ainsi un registre, comme une variété linguistique qui associe à une situation de communication particulière, des caractéristiques linguistiques omniprésentes.

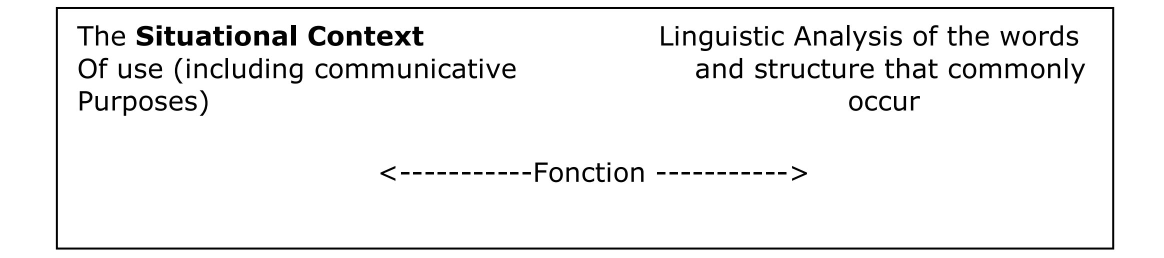
> « *We defined a register as a language variety associated with both a particular situation of use and with pervasive linguistic features that the serve important functions within that situation of use.* » (Biber et Conrad, 2009, p 6)

La démarche de Biber (1988, 1994) base sur une distinction entre trois éléments : 1) une situation de communication, 2) des caractéristiques linguistiques, et 3) une association entre ces deux composantes (cf. figure 6.1). D'après Biber (1988, 1994) les registres sont d'abord décrits à partir de caractéristiques situationnelles. Ces situations peuvent être défini selon :

- Les participants : l'identité des personnes à l'origine de la situation de communication et de la personne à qui le message est adressé.
- la nature des relations entreprises entre les participants;
- les circonstances de production de la communication ;
- les canaux de la communication utilisés.

Chaque situation se distingue par des marques linguistiques telles que la fréquence et la distribution des mots utilisés, des caractéristiques grammaticales du langage (langage simple ou complexe), le degré de leurs utilisations, etc.. Ces marques sont identifiées par leurs fréquences dans le texte. Selon Biber (1994) c'est la variation de l'utilisation de ces marques, qui distingue différents types de textes.

Figure 5.1: Components in a register analysis (Biber et Conrad, 2009, p 6)



D'un point de vue méthodologique, Biber (1988, 1994) se base dans une première étape, sur une procédure d'automatisation, qui lui permet de prendre en compte quantitativement les variations linguistiques dans un corpus de taille importante. Ceci l'amène à déterminer dans une seconde étape, et de manière qualitative, les paramètres situationnels qui fondent ou provoquent cette variation.

I.A.3. Registre des forums électroniques

Les échanges via Internet ont attiré l'attention d'un nombre de chercheurs linguistes ; E-mail, chat, blog ou forum, etc. sont des formes de communication médiatisée par ordinateur (CMO), dans la mesure ou elles permettent à des participants de communiquer en utilisant un ordinateur et le réseau Internet.

Les similarités et les différences entre ces nouvelles formes de langage et les formes du langage oral et écrit ont été un centre d'intérêt pour beaucoup d'auteurs (Murray, 1991; Ferrara et al., 1991; Yates, 1996 ; Ko, 1996). Bien qu'en général le mode de production de la communication médiatisée est l'écrit, ces auteurs montrent qu'il présente des caractères qui se rapprochent des échanges parlés. D'après Ferrara et al. (1991), le langage médiatisé se rapproche plus souvent du langage oral par son caractère informel, simple et spontané, et du langage écrit par sa densité lexicale.

Murray (1991) et Yates (1996) mettent également l'accent sur cette nature hybride du langage médiatisée. En effet, ils observent l'usage fréquent de ce qu'ils appellent de « dispositifs alternatifs » (ponctuation, typographie, etc.), dont le rôle est de combler l'absence des éléments tel que les gestes et les regards, caractéristiques des situations de communication orale.

Murray (1991) remarque également que dans ces environnements médiatisés, les participants ont tendance à adopter un langage simplifié en supprimant les pronoms personnels, utilisant des abréviations de mot courts.

Ko (1996) observe que ces caractéristiques sont encore plus fréquentes, dans des contextes médiatisés tels que les forums où les échanges sont instantanés. Pour faciliter les échanges et être compréhensibles, les participants à ces forums utilisent un langage bref et concis, se rapprochant plus d'un langage oral.

Ainsi, comme nous pouvons le constater, les linguistes identifient plusieurs caractéristiques des communications médiatisées en comparant le langage médiatisée à la dichotomie langage écrit/ oral. Cependant, ces caractéristiques varient selon que les participants échanges de manière synchrone ou asynchrone (Ko, 1996). En effet, Ko (1996) montre que les échanges synchrones se rapprochent plus des échanges oraux, du fait du temps disponible pour le participant pour réagir instantanément. Ceci diffère lorsque les échanges sont asynchrone ; c'est-à-dire lorsque le participant dispose de plus de temps pour préparer son intervention. Ce mode d'échange se rapproche plus des échanges écrits.

Dans le cadre de ce travail, nous nous intéressons à une forme particulière des communications médiatisées : les forums électroniques. Nous présentons dans ce qui suit deux des aspects linguistiques les plus étudiés dans les forums électroniques, à savoir les aspects syntaxiques et les aspects de formes.

I.A.3.a. Aspects syntaxiques

L'analyse des pronoms (Biber et Conrad, 2009) fait apparaître que les forums électroniques sont composés d'un faible pourcentage de pronoms en comparaison avec les e-mails. En effet, dans une étude réalisée sur le forum de discussion des utilisateurs des produits Appel, Biber et Conrad (2009) observent la « suspension» des pronoms subjectifs dans plusieurs des messages échangés (voir exemple de texte 1). D'après eux, ces «suspensions» sont typiques des situations de communication orale, tel que la conversation. Elle est due à la rapidité des interactions et des échanges entre participants.

Exemple de texte 1 : Messages postés sur le forum des utilisateurs des produits Apple

« Worked for me too. [***It*** subject deleted.]

Solved my own problem [***I*** subject deleted]

Any ideas? *[**Do you have*** subject + verb deleted] » (Biber et Conrad, 2009, p

Les résultats mettent aussi en avant que comparativement aux e-mails, le pourcentage les pronoms de troisième personne, sont plus importants dans les messages postés dans les forums électroniques. En effet, selon Biber et Conrad (2009) un e-mail est adressé personnellement à son destinataire, tandis que les messages postés dans les forums sont destinés à l'ensemble, et identifiés par le collectif. Ceci explique la fréquence des pronoms tels que « ***it, they, them,*** etc ..» dans le texte étudié.

L'analyse des aspects syntaxiques montre une forte proportion de pronoms de troisième personne dans les forums électroniques, qui tendent à rapprocher celui-ci du langage écrit. Cependant ces proportions changent, lorsqu'on compare l'usage des pronoms entre « novices » et « experst » (Biber et Conrad, 2009). C'est ainsi que dans le même contexte de l'étude, Biber et Conrad (2009) observent que les pronoms de première et deuxième personne, sont utilisés de façon similaire par les « experts » et les « novices ». A l'inverse, les messages en provenances des experts, incluent un fort pourcentage de pronoms de troisième personne. D'après les auteurs, « novices » et « experts » participent au forum dans le but de partager leurs expériences, sur un sujet bien spécifique. Dans le cas étudié le forum regroupe des utilisateurs de produits Apple, et les échanges concernent les problèmes rencontrés au moment de l'utilisation ou de l'installation d'un produit déterminé. Ceci qui justifie la fréquente utilisation du pronom personnel « ***I*** » chez les novices (***I*** spent all night greting it astivated …and now ***i*** can revive call) ainsi que chez les experts (***I*** went to bed, ***I*** woke up, ***I*** picked it up, ***I*** put it). Les résultats sur les pronoms de troisième personne, mettent en avant le fait que les « experts », contribuent au forum pour donner des explications, des instructions ou plus de détails. Des marqueurs tels que « ***she*** told me », « ***they*** had closed », « customer service can help ***them*** », « because ***it*** upset me greatly » sont ainsi plus fréquents dans les messages postés par les « experts ».

Les pronoms de deuxième personne sont beaucoup moins fréquents chez les « experts » en comparaison aux premiers et aux troisièmes, et sont utilisés occasionnellement, pour donner des conseils aux utilisateurs. Ko (1996) suggère que la présence de pronoms de deuxième personne correspond à une substitution de l'absence d'interaction non verbale par des pronoms impliquant le destinataire.

Concernant les verbes et adverbes, les analyses dévoilent que leurs usages sont assez fréquents dans les forums. Biber et Conrad (2009) observent en effet, que les messages postés dans les forums sont de types descriptifs, courts et centrés sur un sujet ou un problème spécifique.

L'utilisation des adverbes est donc très fréquente dans ce type de messages. Par ces messages, les participants au forum décrivent des problèmes rencontrés (par exemple: « *When i run safari ipod features, the iphone goes into landscape automatically, still waiting* » (Biber et Conrad, 2009, p 195)), ou proposent des plus d'instruction et d'explication concernant un problème (par exemple : « *The iphone requires both 10.4.10 and iTunes 7.3 to function* ***correctly,*** *If you use a Mac with the iphone i twill work* ***perfectly*** » Biber et Conrad, 2009, p 195)). Biber et Conrad (2009) observent également que dans certains messages, les utilisateurs expriment leur excitation ou désapprobation, a propos du produit, ici l'i-phone. Leurs échanges se caractérisent par la présence d'adverbes tels que : « ***Tottaly*** », « ***absolutly*** », « ***incredibly*** », qui correspondent à l'expression d'émotions.

Sur l'utilisation des verbes, les auteurs constatent que dans un nombre important de messages postés sur le forum, les verbes sont utilisés par les participants dans le but de demander conseils auprès des utilisateurs experts. Cette utilisation est souvent précédée par une description du problème rencontré et des tentatives pour le résoudre. L'exemple suivant (exemple texte 2) illustre une forte utilisation des verbes dans les messages émis par les « novices ».

Exemple de texte 2 : Messages postés sur le forum des utilisateurs Apple

« …whilst **charging I cannot get** the thing to **start, go** on, **reset. HELP!**

Yes, **I have** itunes 7.**3 installed.** I can't get tit too **mount. I have restarted, reinstalled** 7.3 and **unplugged** it and **replugged** it a handful of times

…any other ideas? » (Biber et Conrad, 2009, p 195)

I.A.3.b. Aspects de forme et d'expression

Les messages émis dans les forums sont écrits, cependant, ces messages présentent des particularités par rapport aux autres types de textes écrits (e-mails, lettres, articles scientifiques, etc.).

D'abord, au niveau de la présentation, les marques d'ouverture et de clôture sont absentes ou fortement réduites, dans les messages des participants aux forums. En effet, Biber et Conrad (2009) observent que l'utilisation des prénoms, des signatures d'ouverture ou de clôture que « ***hi, dear, etc.*** » est très faible (seulement une dizaine de messages dans le forum Apple). Panckhurst (1999) observe les mêmes comportements dans les e-mails ; dans une étude portée sur 115 e-mails échangés entre les étudiants d'une même classe, il montre que 19 de ces e-mails ne contiennent aucune formule d'ouverture, et que les formules de salutation ou de clôture sont absentes de 32 messages.

Généralement placés en fin de phrase (Marcoccia, 2000), les simleys sont des marques très fréquentes dans les échanges médiatisées. Ils servent à travers des signes :-), ;-), :->, :-(, :-O, : [ ][60] à exprimer un sentiment, signaler une attitude ou une émotion.

L'émotion peut être également exprimée à travers l'utilisation de la « ponctuation expressive ». Selon Crystal (2001), dans les forums, l'expression se traduit par la répétition de caractères tels que (hiiiii, sooo, !!!!!, ????) ou l'utilisation de la majuscule (NO MORE !!!). Le rôle de ces marques, est de signaler l'importance des propos ou d'attirer l'attention des internautes. Crystal (2001) observe également que les parenthèses, les points d'exclamation, d'interrogation sont très utilisée dans de tels contextes.

Ces études suggèrent ainsi que les forums sur Internet, correspondent à un registre de langage qui les distingue des autres formes de communications (écrit, parlé, ect.).

Nous avons également constaté qu'à l'intérieur d'un registre, des genres de langage peuvent se distinguer par des marques spécifiques.

Si nous considérons que l'organisation de l'activité au sein de Bugzilla est fondée sur la communication entre plusieurs participants, nous pouvons prétendre que le langage utilisé

[60] Marcoccia (2000) définis six smileys les plus courants sur le net : :-) heureux, ;-) clin d'oeil, :-> sourir sarcastique, :-(triste,:-o surprise,, :[ ] Crier.

dans ce contexte présente un genre particulier reflétant des aspects organisationnels spécifiques.

I.B Procédure pour étudier le langage dans Bugzilla

Dans cette partie, nous nous intéressons aux aspects méthodologiques et techniques de la mise en place d'une analyse linguistique, basée sur l'identification de genres ou des caractéristiques linguistiques du langage utilisé au sein de Bugzilla. La procédure consiste en quatre étapes : sélection, annotation et lemmatisation, traitement automatique, et analyse (cf. figure 5.3).

I.B.1. Sélection du corpus de l'analyse

Dans le cadre notre recherche, nous nous ne donnerons pas une définition stricte du terme corpus. Nous le présenterons d'avantage comme un échantillon de données linguistique que nous souhaitons analyser, dans le but d'observer des phénomènes linguistiques et ce, indépendamment de théories linguistiques précises.

Le corpus à analyser est composées de textes qui sont présentés sous forme d'un rapport, contenant plusieurs messages. Notre corpus est ainsi constitué d'une sélection de rapports de bogues composés chacun de plusieurs messages.

Dans le cadre de cette étude exploratoire, notre objectif est ainsi d'observer un phénomène nouveau. Ce qui explique que notre échantillon n'était pas entièrement spécifié, mais a évolué au cours de l'analyse. Au terme de ce travail, nous avons constitué un corpus de 649 rapports de bogues.

I.B.2. Annotation et lemmatisation du corpus

Pour notre part, le corpus retenu ne sera pas « annoté [61]» comme il l'est dans certains cadres retenus en linguistique, on étudiera le texte, grâce à des outils automatiques. Autrement dit, le texte électronique en entrée ne subit aucune manipulation manuelle

[61] Les annotations sont « des repères ou commentaires rajoutés dans le texte. Elles sont reconnues par le logiciel et permettent de créer des variables de contexte caractérisant les fragments auxquels elles s'appliquent. » (Analyse Lexicale avec Le Sphinx Manuel d'utilisation, 2006)

d’annotation, avant que n’interviennent les stades de lemmatisation[62] et d’analyse automatisés.

L’annotation interviendra dans une deuxième phase de l’étude (cf. chapitre VI), son rôle étant de définir des unités élémentaires, qui serviront à découper le corpus pour les comparer dans l’objectif de procéder à une analyse contextuelle plus fine.

I.B.3. Traitement automatique du corpus

Pour étudier le corpus nous l’avons soumis à l’outil d’analyse textuel Sphinx, qui est un logiciel analysant des documents en format de texte. Il assure automatiquement le découpage du texte en unités lexico-syntaxique et sa lemmatisation.

Notre objectif était de dégager une répartition concernant les caractéristiques linguistiques employées. Suite à cela nous avons obtenu un corpus composé des traits linguistiques présentés dans le tableau suivant (tableau 5.2)

Tableau 5.2 : Unités lexico-syntaxique du corpus

Unités lexico-syntaxique	Unités (occurrences)
Corpus (692 rapports de bug)	750557
Verbes	151298
Noms	185119
Adjectifs	47029
Marque d'émotion	430
Pronoms	68958
Adverbes	11848
Conjonctions	33796
Prépositions	68505

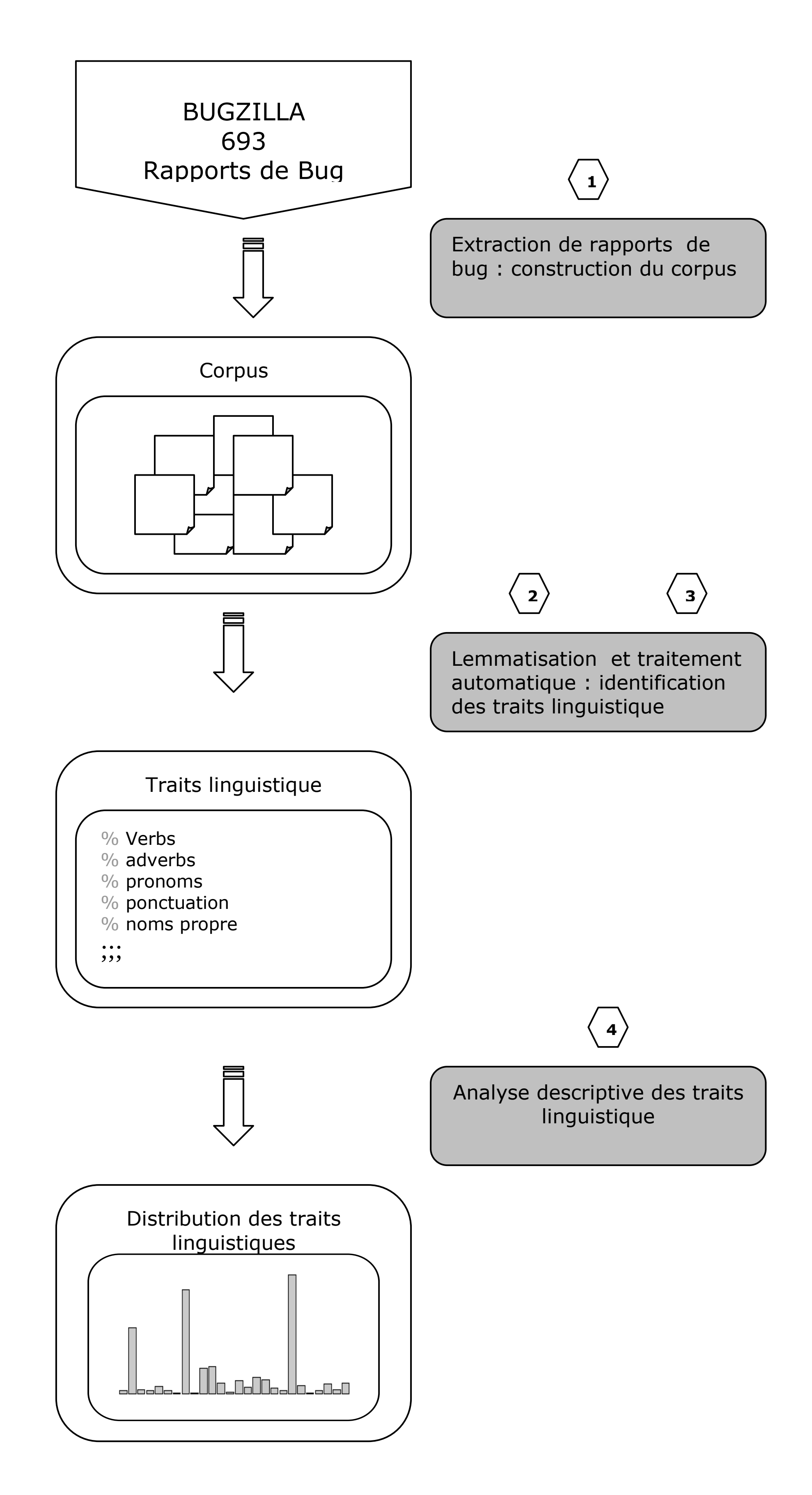
[62] La lemmatisation se définit comme « une méthode qui permet de ramener chaque mot à sa forme racine: l’infinitif des verbes, le masculin singulier des noms et adjectifs. Enfin, présenter le lexique par catégories grammaticales permet de focaliser l’attention sur les objets (substantifs), les actions (verbes) et évaluations (adjectifs). » (Analyse Lexicale avec Le Sphinx Manuel d'utilisation, 2006)

I.B.4 Analyse descriptive des traits linguistiques

La dernière étape de l'analyse consiste à interpréter les résultats obtenus, avec les éléments suivants d'interprétations :

- *Quelles sont les propriétés du langage utilisé dans le cadre de Bugzilla ?*
- *Y a –t-il des spécificités d'ordre linguistique et extra-linguistique, dans le texte qui font apparaitre un genre de langage spécifique à Bugzilla ?*
- *Ce genre reflète –il certaines propriétés explicitant l'organisation de l'activité au sein de Bugzilla? et permet par conséquent de définir les fonctions de la communication dans l'organisation de cette activité ?*

Figure 5.2 : Vue d'ensemble de la procédure de l'analyse



Section 2. Analyses et résultats

Cette section est consacrée à la présentation des résultats issus de l'analyse du langage utilisé au sein de Bugzilla dans l'optique d'identifier ses différentes propriétés en rapport avec l'organisation de l'activité de résolution de bogues.

Nous abordons dans un premier temps, une analyse détaillée d'un exemple de rapport de bogues (le rapport : BR-362919) dans l'objectif d'apporter un premier éclairage sur les aspects qui caractérisent la résolution de problèmes dans Bugzilla.

Nous exposons dans un second temps, les résultats issus de l'analyse linguistique des 692 rapports de bogues sélectionnés.

I.A. Un exemple : Analyse du Rapport (BR-362919)

Pour analyser le rapport BR-362919, nous avons choisi d'utiliser la méthode développée par Strauss et Corbin (1990). Cette méthode consiste à étudier les données textuelles de manière inductive en les examinant ligne après ligne. Le rapport (BR-362919) est présenté et analysé de la manière suivante :

- Le rapport sera d'abord présenté sous forme de tableau (cf. annexeV.1) :
  Dans ce tableau figure à la première colonne le « *bug activity* ». La deuxième colonne présente les messages échangés entre les développeurs correspondant à chaque changement dans l'activité (le changement du statut du bogue, de priorité, de sévérité, de l'identité du responsable du bogue, etc.). En effet, le « *bug activiy* » fourni un aperçu général de l'évolution de la résolution du problème. Il permet également d'identifier l'identité des contributeurs (la personne que la déclaré le bogue « *bug reporter* », le responsable du bogue « *bug assignee* », etc.).
- Dans un seconde étape nous donnons une vue d'ensemble des caractéristiques de la résolution (cf. figures 5.4). Ceci en représentant l'évolution des échanges entre participants et de l'activité de la résolution en général (le « *bug activity* »).

Nous signalons que l'objectif d'une telle présentation est d'avoir une correspondance entre les échanges entre développeurs et l'évolution de l'activité de la résolution du bogue.

Pour des raisons de lisibilité le format original du rapport de bogue et du « bug activity » sont présentés en annexe V.2 et V.3.

II.A.1. Description du rapport «BR-362919»

Le tableau 4.1 contient la totalité des messages postés dans le rapport de bug (BR-362918). Ce rapport est sélectionné à partir de Bugzilla sur le site www.bugzilla.mozilla.com. Notre choix d'étudier en détail ce rapport est lié au fait qu'il présente plusieurs phénomènes typiques de Bugzilla.

Ce rapport indique que la résolution commence par la déclaration et la description du problème signalé. Le rapporteur donne une première description du problème. Il présente au départ des informations concernant son environnement d'utilisation (information sur des faits) (#Description), ensuite un lien qui permet de voir le comportement problématique (Comment #1 et Comment #2). D'autre contributeurs réagissent pour décrire une solution (Comment # 3) puis fournir plus d'informations sur le problème (Comment#4 et #5). On constate également des échanges entre le rapporteur et les autres contributeurs pour la description du problème (Comment#6 et #7).

Les messages de description permettent donc de fournir un maximum d'informations au collectif pour, orienter la résolution du problème.

Lors de cette phase de description et de définition, les échanges se caractérisent par :

- un transfert d'informations techniques permettant de reproduire le problème ou de le rapprocher d'un bug similaire (bug duplicates).

Comment #1 From Tim Maks van den Broek 2006-12-06 01:16:48 PDT

Created an attachment (id=247651) [details]

Comment #9 From Magnus Melin 2006-12-18 08:33:04 PDT

*** ~~Bug 364211~~ has been marked as a duplicate of this bug. ***

- des échanges d'informations sur des faits ou des informations spécifiques sur l'environnement du problème.

Description From Tim Maks van den Broek 2006-12-06 01:10:17 PDT

When i use the preview pane in build version 2 beta 1 (20061205) the pane is to wide. The button "not scam" / show images are off screen. (see screenshot)

Comment #4 From Mike Cowperthwaite 2006-12-06 10:41:24 PDT

In TB 2, the alert bar no longer wraps its text when it's made narrow; it has aminimum width which (in the English version, anyway)

```
is wide enough for the text "To protect your privacy..." plus the
icon and the Load Images button.  If the window is narrowed below
that width, the window edge will start to encroach on the bar (and on
the message body, too, which never gets narrower than the alert bar,
if there is one).
```

Les autres messages concernent la résolution du problème. En effet, une fois le problème décrit, plusieurs versions de solutions ou patches ont été émis (Comment#10, #11), testées (Comment #14, #16), révisés (Comment #12, #17) et le problème enfin vérifié et fixé (Comment #18, #19, #20).

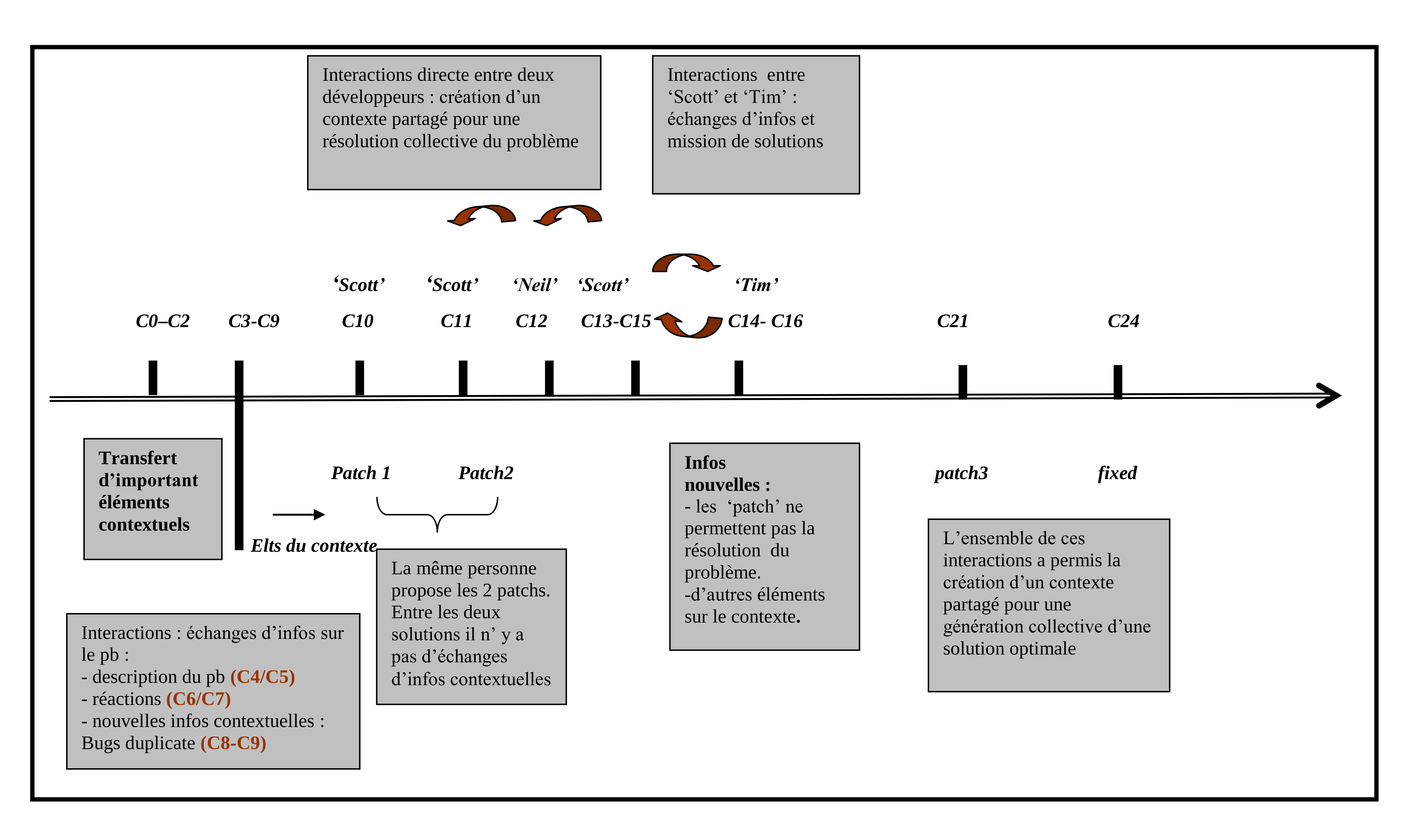
Dans cet exemple, la résolution débute par des propositions de solution (l'émission de patchs) ; par exemple dans les commentaires #10 et #11, *Scott* propose deux versions de patches. Il donne également une description des solutions proposées, ainsi qu'un lien qui contient le patch.

Il s'adresse ensuite directement à *Neil* et *Tim* (le rapporteur du bug) pour réviser et tester les solutions proposées. Une fois cette solution est testée par *Tim* et révisée par *Neil*, ce dernier fixe le bogue en proposant une dernière version révisée du patch.

A partir de cet exemple nous pouvons remarquer que la résolution du bogue se compose de cinq étapes : la description du problème, la proposition de solution, la révision, le test, et la fixation. A cette phase de résolution, les échanges se caractérisent par :

- La description de la solution (Comment #10, #11) ;
- des demandes d'exécutions d'actions (Comment #10, #11, #13) ;
- des interactions directes entre développeurs (Comment#10, #11, #12, #13, #15).

Figure 5.3: Vue d'ensemble des échanges pour la résolution du bug BR-362919



II.A.2. Synthèse sur l'analyse du rapport (BR-362919)

L'analyse du bug (BR-362919) montre qu'un nombre limité de développeurs (deux participants) contribuent à la construction de solutions au problème (l'émission de patches). Pour ce cas, c'est le responsable du bug qui émet des patchs en interaction avec un autre participant (voir figure 5.3).

Ces constatations rejoignent les travaux de Mockus et al. (2002) qui développent, en étudiant le projet Apache, l'hypothèse de la centralisation du développement : « *seulement 15 développeurs contribuent à 80% de l'écriture du code* ».

Dans cet exemple, l'activité de résolution est composée de quatre phases : *la définition de problème, la construction de solutions, la révision et le test de la solution et la fixation du problème*.

Nous identifions ainsi à partir des commentaires, quatre propriétés, caractérisant les échanges entre les contributeurs (cf. figure 5.3):

- *L'explicitation de l'action :* dans certains messages les contributeurs déclarent explicitement qu'ils ont réalisé une action ;
- *La description* : ces échanges se caractérisent par un transfert d'informations spécifiques sur l'environnement du problème, mais aussi, des informations techniques permettant de reproduire le problème ou de l'identifier comme un problème identique à un autre « *bug duplicates* ».
- *Les directives* : dans ce cas, les échanges sont directement liée à une action par une demande de réalisation « `Tim, ....Can you test today's; asking Neil for a review` »,
- Des interactions directs entre les participants :« `Tim, …Can you test……. ?`», « `Rhigt,: ))` », « `it is a lot better but still not ok` », « `Hey, …….;-)…….. :-)` ».

Le rapport analysé peut être considéré comme typique, dans la mesure où il a permis d'identifier et de déterminer certaines caractéristiques de l'organisation de la résolution de problème. Toutefois, à partir d'un seul exemple nous ne pouvons pas définir toutes les caractéristiques. En effet, comme nous l'avons présenté dans la première section, Bugzilla est un environnement très diversifié, il faut donc pour pouvoir le caractériser étudier plusieurs

rapports de bogues plus significatifs. C'est ce que nous nous proposons de faire dans la partie suivante à partir d'une sélection de 694 rapports de bogues.

II. B. Caractéristiques générales du langage dans Bugzilla

En analysant les résultats de notre corpus élaborés par Sphinx, nous étions à la recherche de marques explicites, qui indiquent le genre du langage utilisé dans le cadre de Bugzilla. Pour cela, nous nous sommes attachés dans un premiers temps à identifier les traits linguistiques les plus employés. Les résultats montrent que notre corpus composé de 692 rapports de bug comporte 750 557 occurrences et 26717 formes uniques.

La figure 5.7 présente la distribution des fréquences de 8 catégories linguistiques, fournies par le logiciel Sphinx, à partir de notre corpus. Pour chaque catégorie linguistique, nous avons déterminé le pourcentage des occurrences identifiées par le logiciel.

Dans le cadre de cette analyse nous n'évoquons que les verbes, les noms, les adjectifs, les adverbes, les pronoms, les marques expressives (signes d'émotion, ponctuation, etc.), les conjonctions et prépositions.

A partir de ce graphique, nous pouvons constater que le genre de Bugzilla se démarque nettement au niveau de l'utilisation des noms et des verbes (qui représentent 44,1 % de l'ensemble des occurrences).

La distribution des fréquences révèlent également un emploi moins considérable de prépositions (9,18%), de pronoms (9,12%), d'adjectifs (6.26%), et de conjonctions (4,50%).

Ces résultats semblent concorder avec d'autres travaux ; en effet, comme nous l'avons déjà présenté, Biber et Conrad (2009) observent dans une étude sur le forum des utilisateurs des produits Apple, une forte utilisation de verbes et un faible pourcentage de pronoms personnels.

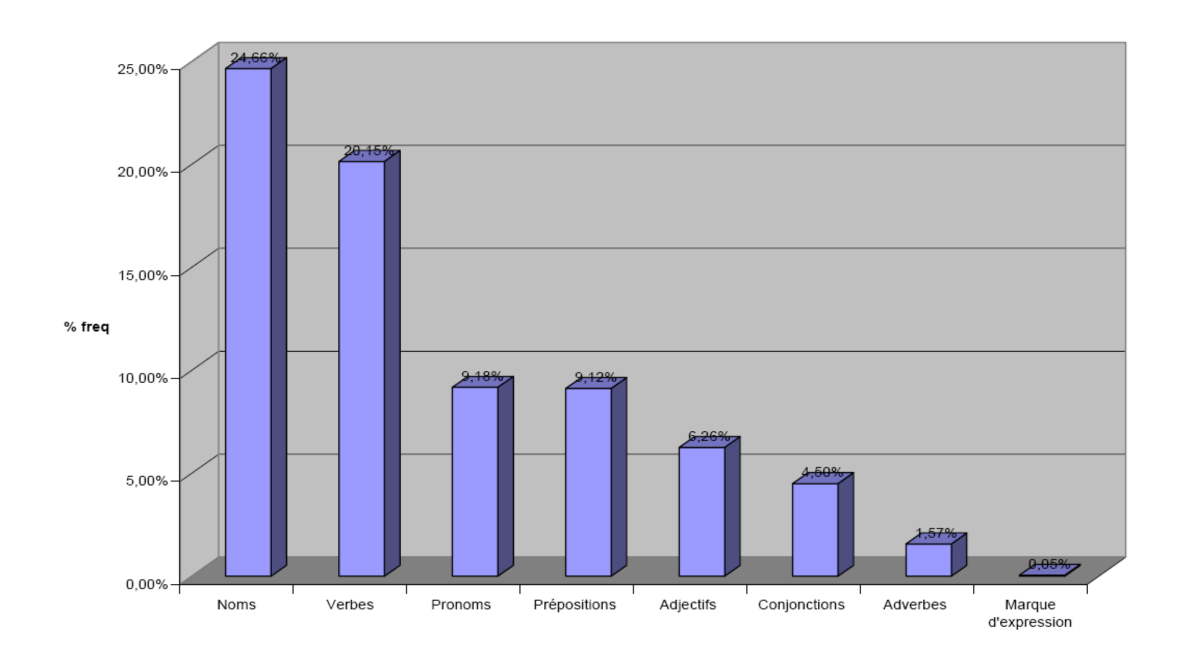
Sur l'usage des noms, Andrew et al. (2006) constatent que sa proportion est très élevée (66%) dans des contextes comme Apache, Open Office et Eclipse. Ils rajoutent que leurs rôles est principalement descriptif.

On remarque cependant, un déficit dans l'usage d'adverbes (1.57%) et de marques d'expressions (0.05%).

Ces dernières constatations divergent avec ce que les travaux en linguistiques (Biber, 2003; Crystal, 2001) ont pu observer dans des contextes où les échanges sont principalement médiatisés et qui se distinguent selon eux par la forte présence de ces marques expressives.

Nous avons jusqu'à présent observé les catégories linguistiques de manière générale, nous nous proposons dans une seconde phase d'approfondir l'analyse pour l'examen de chacune de ces catégories, afin de mettre en évidence le genre du langage étudié.
Dans la mesure où les catégories, marques d'expression, adverbes et conjonctions présentent des proportions faibles, nous examinons seulement les catégories noms, verbes, adjectifs, pronoms et prépositions. Ces dernières seront caractérisées de manière descriptive.
Nous rappelons que notre étude du langage vise essentiellement à identifier le genre du corpus de manière qualitative sans tenter d'identifier de phénomènes linguistiques comme la manifestation de l'énonciateur dans le texte, les fonctions des traits linguistiques dans le texte, etc.

Figure 5.4 Distribution des catégories linguistiques au sein du corpus (en% des occurrences)



I.B.1. Les noms

Les résultats de l'analyse syntaxique montrent clairement l'importance de l'utilisation des noms (24,66%). En effet, dans le corpus étudié, nous avons identifié pour cette catégorie 8171 formes (30,83% du nombre total des formes).
Pour avoir plus de compréhension sur les caractéristiques de ces formes, nous avons limité nos observations, pour des raisons techniques, aux formes les plus fréquentes.

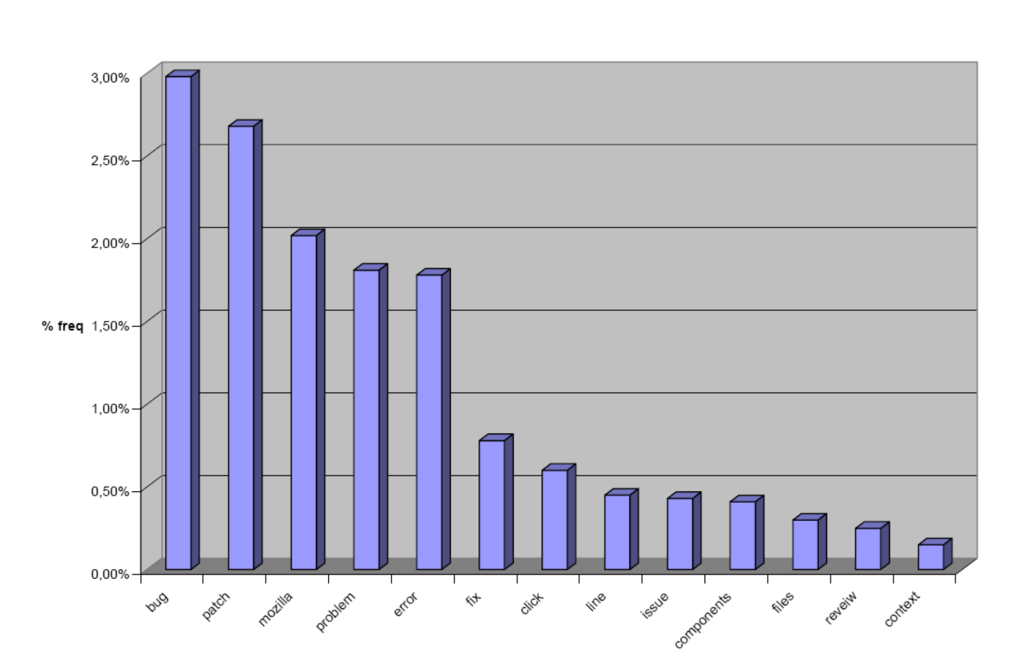
Comme l'illustre la figure 5.5, les noms les plus fréquents dans le corpus désignent le problème rencontré «*error* », « *bug* », « *problem* », etc. On relève également un emploi considérable de noms comme «*file*», « *context menu*», « *component* », « *line* ».

Ces résultats montrent que les développeurs se réfèrent souvent a des entités visuelles, pour décrire les problèmes. Par exemple, les rapporteurs d'un problème, donnent souvent une description de la situation rencontrée ou renvoient directement les développeurs vers des liens, qui permettent de visualiser ou reproduire ce problème : « *The **error** console: **Error : #UFILE** has no properties Source file : chrome : / / browser / **content** / bookmarks / bookmarks. js Line : 103 - Error : Components* » ;« *Vaguely (ir) relevant : In 4 . x for Windows the back / forward **context menu** could be obtained either by right* ».

Parmi les formes les plus fréquentes, nous trouvons les noms qui désignent une action, comme par exemple la révision du problème («review=*me* ») ou sa fixation (« *The actual **fix** for this particular problem IIRC was removing several lines of code in nsCSSFrameConstructor* »).

Nous pouvons déduire de ces résultats, que le rôle de l'utilisation des noms dans le cadre de Bugzilla est principalement descriptif. Ceci peut s'expliquer par le fait que dans le contexte de résolution de problèmes dans lequel nous nous plaçons, les participants doivent fournir le plus d'informations sur le problème et son contexte, afin que les participants puissent construire une connaissance collective du problème, de ses symptômes et produire à leur tour des informations utiles à sa résolution.

Figure 5.5 Distribution des noms les plus fréquents au sein du corpus (en%)



I.B.2 Les verbes

Dans le corpus global nous répertorions 6368 différents verbes (23,83% du nombre total des formes), les verbes les plus fréquents étant présentés dans le tableau 5.6[63]. Selon Bouscaren et al. (1987), les formes verbales se classent en deux catégories : les formes simples et auxiliés.

Les formes simples indiquent que l'énonciateur livre une information, qui semble indépendante de tout point de vue particulier.

Les formes auxiliés au contraire indiquent que l'énonciateur prend position par rapport au contenu qu'il propose.

A partir des résultats, nous pouvons constater que c'est dans cette dernière catégorie, que la majorité des verbes utilisés dans le corpus, se classent. En effet, les verbes sont souvent accompagnés d'auxiliaires de modalité comme do, shall, will, must, etc. L'information livrée se base donc, sur l'expérience et le point de vue de son énonciateur.

Les participants utilisent ainsi ces verbes pour :

- décrire certains aspects problématiques du logiciel («*In the suite there* **is** *no cancel button* »);
- demander explicitement à un participant d'accomplir une quelconque action (*«* ***add*** *this to usercontent.* »);
- exprimer un désir pour réaliser des tâches spécifiques (« *pressing a Finder icon* ***should*** *select the icon but not open it* ») ;
- donner des instructions pour l'exécution de certaines taches (« open », « bild », « show », « clic », « instal », « create », etc.).
- signale la réalisation d'une action (« I **applied** the patch », « ***verified*** *code checkin* »).

L'analyse des aspects syntaxiques, montre que Bugzilla représente un contexte dans lequel les participants décrivent le problème, en effet le rôle du verbe est d'exprimer l'action subie par le problème.

Ces participants interviennent également en donnant des instructions et en fixant les problèmes. Le verbe permet, dans ce cas, d'exprimer l'action accomplie pour résoudre ces problèmes. Nous pouvons en outre constater que ces actions, marquent certaines étapes du processus de résolution, comme l'affectation du problème à un responsable, la révision, la vérification et la fixation du problème.

[63] Ces verbes comptent pour 36,65% de l'ensemble des verbes utilisés dans le corpus.

Tableau 5.3 Catégories des verbes les plus fréquents

Verbes	Fréquences[64]	Exemples
is	12,35%	The checkbox **is** not saved.
have	3,07%	I **have** still no idea what ' s going wrong
will	3,03%	It appears as if releasing the button **will** do nothing.
can	2,44%	I **can** confirm this on Firebird too.
do	2,44%	Dean if you do decide to **do** this sooner feel free to retarget.
use	1,89%	If you **use** the Manage menu item you will see all your
Get	1,68%	You will **get** streams of regression bugs that context
Click	1,26%	Double - **click** text in the location bar to select it Mozilla / 5 . 0 (Windows ; U ; Windows NT 5 . 1 ; en - GB
fix	1,22%	**fix** that and this bug goes away
think	1,22%	I **think** the current implementation requires you to check the box.
go	1,13%	**go** to url 3 . download installer . exe Actual Results : Directory with same file name.
See	1,05%	**see** Bug 242599 as that would be the best place for your work .
make	1,01%	**make** it listen to command rather than click
Total	*34,34%*	

I.B.3. Les pronoms personnels

Même si les pronoms personnels présentent, un pourcentage moins considérable (seulement 9,18%) en comparaison avec les noms et les verbes, il est cependant important de caractériser le genre du langage utilisé dans Bugzilla, par rapport à l'utilisation cette catégorie linguistique.

Comme l'illustre la figure 5.7 suivante, rassemblant les résultats obtenus à partir de l'analyseur Sphinx, les pronoms de deuxième personne sont les plus marginaux (12.7% des pronoms personnels). Nous pouvons donc constater que les échanges directs entre les participants ne sont pas très fréquents. Ceci explique le fait que, dans le cadre de Bugzilla, l'information transmise ne concerne pas une seule personne mais l'ensemble des participants. Les pronoms de première et troisième personne, sont les plus employés dans le corpus (49,42% des pronoms personnels sont de première personne et 37,73% sont de troisième personne).

[64] Fréquences par rapport au nombre d'occurrences des verbes dans le corpus (68958)

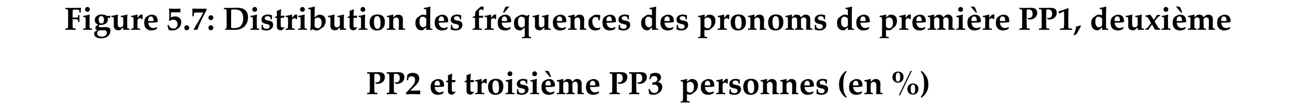
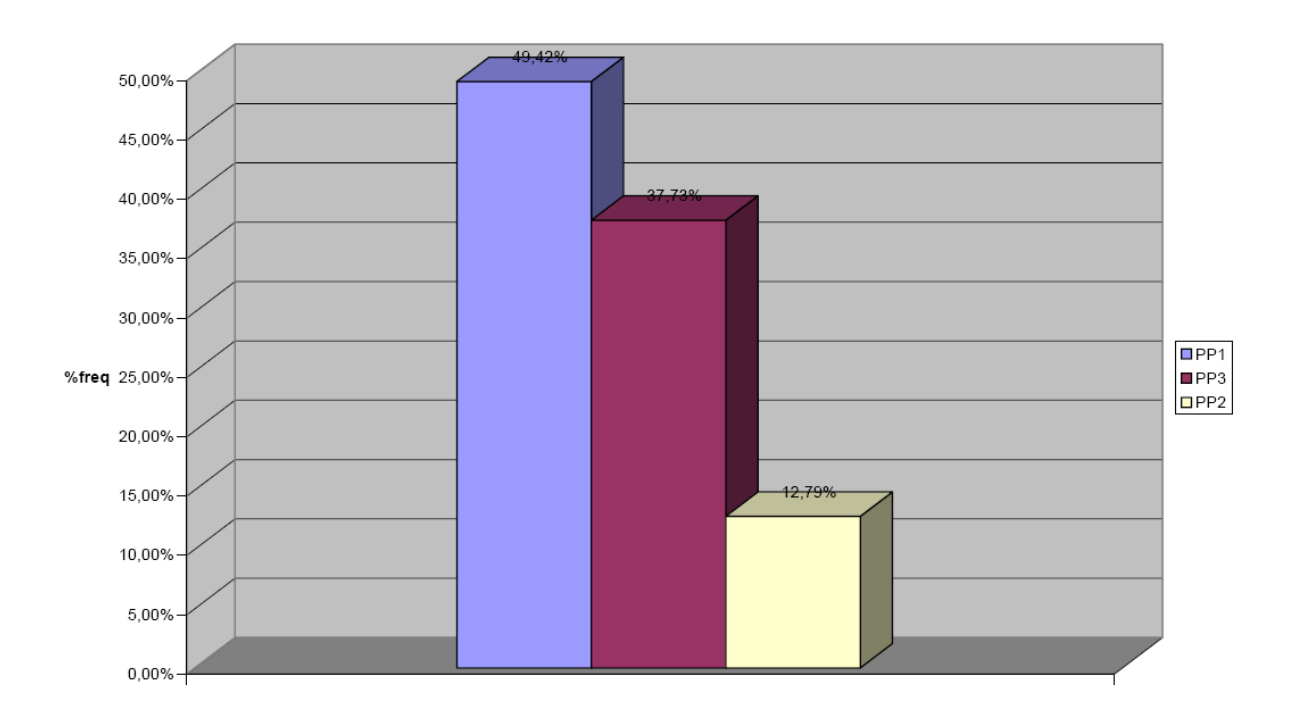


Figure 5.7: Distribution des fréquences des pronoms de première PP1, deuxième PP2 et troisième PP3 personnes (en %)

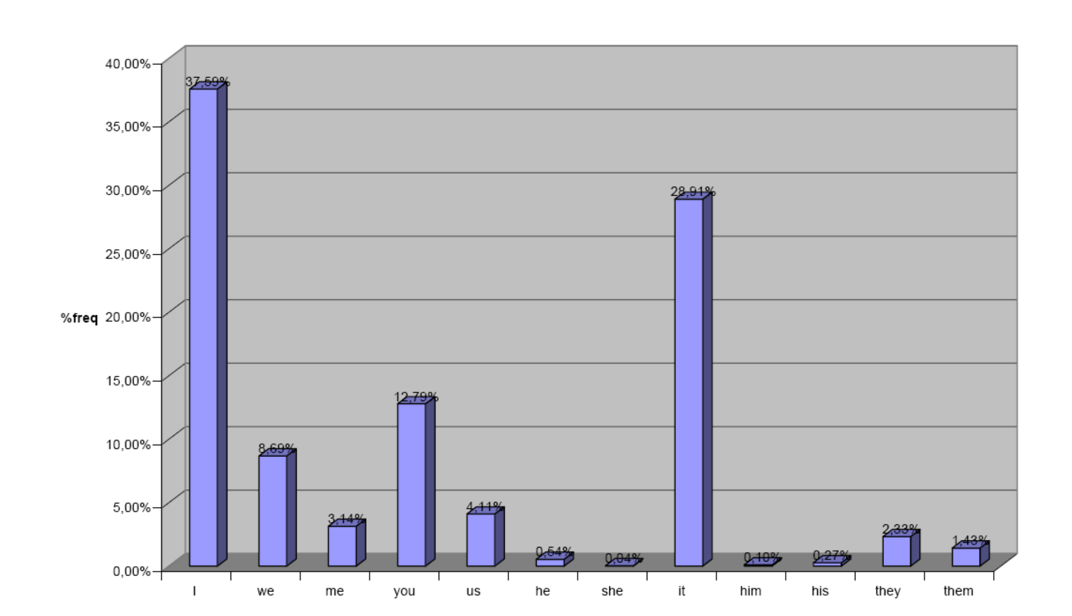


La figure 5.8 indique que les pronoms de première et troisième personne du singulier, sont fortement utilisés par les participants. Nous constatons que le pronom « ***I*** » est particulièrement représenté dans le genre de Bugzilla (37,59% de l'ensemble des pronoms personnels employés).

Le pronom de première personne du pluriel, est toutefois moins fréquent dans le corpus (8,69%). Nous pouvons déduire, que dans Bugzilla, les participants tentent de s'investir dans ce qu'ils énoncent plus que d'inclure les autres participants. Dans le corpus étudié, le pronom « ***I*** » est souvent utilisé pour exprimer un besoin « ***I need*** », la possibilité « ***I can*** » ou une pensé « ***I think*** ».

Le second pronom personnel le plus employé est le pronom impersonnel « ***It*** ». Souvent suivit des verbes falloir ou devoir, il permet à sont énonciateur de donner des instructions concernant une situation donnée. De la même manière que « ***I*** », le pronom « ***It*** », est souvent utilisé par les participants pour exprimer l'obligation « ***it should be*** », le besoin « ***it need*** » ou la possibilité « ***it can*** ».

Figure 5.8 Distribution des fréquences des pronoms personnels (en %)



I.B.4 Les prépositions

9,12% des mots utilisés dans le corpus étudié sont des prépositions. En examinant le corpus, nous observons que les prépositions servent à attribuer des propriétés contextuelles comme la position, la source, le type, l'échelle, etc.

Le tableau 5.4 présente les prépositions les plus employées dans le corpus. A partir de ces exemples, nous pouvons constater que le rôle de ces prépositions est principalement descriptif. En effet, en utilisant les prépositions « ***in*** », « ***on*** », « ***of*** », « ***by*** », les participants indiquent l'élément qui a causé le problème. Quand aux prepositions « ***from*** », « ***under*** » et « ***at*** » permettent aux participants de localiser le problème visuellement.

Tableau 5.4: Catégories des prépositions les plus fréquentes

Prépositions	%[65]	Exemple
in	18,63	Screenshot of the error **in** action
of	10,78	I ' m not sure how old **of** a version of Firebird Help that I attached in the patch
on	8,72	There' s also one serious issue with error pages **on** my browser
at	2,65	syntax error **at** line 125
by	2,38	Bug fixed **by** the checkin for bug 240129
from	3,3	Another problem with the current fix is that selecting Zoom **from** the menu twice
under	1,2	Actual Results: very big toolbar **under** the statusbar

I.B.5 Les adjectifs

A la différence des noms et verbes, les adjectifs ne sont pas très employés dans notre corpus. Seulement 6,26% d'adjectifs sont utilisés dans le corpus global. Parmi ceux ci nous distinguons 3657 formes uniques (13,68% de l'ensemble des pronoms personnels employés).

Les résultats présentés dans le tableau 5.5 indiquent toutefois que 6 formes présentent à elles seules 26,82% d'occurrences[66]. L'adjectif « duplicate » est particulièrement représenté dans le corpus (13,2% de l'ensemble des adjectifs employés). L'utilisation de cet adjectif permet de marquer le problème comme « ***duplicate*** » c'est-à-dire identique à un autre problème. La fonction de cet adjectif est importante car elle permet d'orienter la résolution du problème puisque le statut du problème passe à « *resolved duplicates* », si le bug similaire a été résolu.

L'adjectif « ***major*** » à également la même fonction, puisqu'il permet de signaler la priorité du problème en le déclarant comme majeur. La fonction des adjectifs comme « ***no*** », « ***any*** », « ***compatible*** », « ***different*** » et « ***standard*** », est aussi importante puisque, en s'associant au nom, ils permettent de décrire la qualité du logiciel : signaler des différences ou incompatibilités associés à son l'utilisation.

[65] Fréquences par rapport au nombre d'occurrences des prépositions (68505 occurrences)

[66] Frequence par rapport au nombre d'occurrence des adjectifs (47029 occurrences)

Tableau 5.5: Catégories des adjectifs les plus fréquents

Adjectifs	Fréquences	Exemples
duplicate	13,12%	This bug has been marked as a **duplicate** of 134260
major	4,76%	I think this bug should really be **major** as well and the Target Milestone
no	3,62%	In the suite there is **no** cancel button
any	2,08%	I ' m not seeing **any** jar manifests here
compatible	1,08%	it is not **compatible** with your Firefox build
different	1,08%	the patch which tries to use a **different** mechanism to display copyright information
standard	1,08%	Is not **standard** in Mac OS
Total	*26,82%*	

I.C Identification des fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla

Dans la partie précédente, nous avons étudié le langage utilisé dans Bugzilla et identifié plusieurs aspects caractérisant l'organisation de l'activité au sein de Bugzilla. Nous présenterons dans cette partie quatre fonctions de la communication aspects : la spécification des tâches, la définition des rôles, l'explicitation de l'activité, et l'articulation de l'activité

I.C.1 Spécifier les tâches

Les proportions considérables de noms, de pronoms personnels (de première et seconde personnes) montrent que la communication dans Bugzilla est principalement *descriptive.* Dans ce contexte, la description joue un rôle central car elle permet de fournir un maximum d'informations au collectif afin d'orienter la résolution du problème. Les participants contribuent à l'activité en proposant des informations qui permettent :

- de décrire le contexte du problème: décrire les états provoqués par le problème, et l'environnement dans lequel il se situe (par exemple : « *frequently the ad server that is used by one of the forums I read is down, and when the request for the ad fails, Firebird shows a modal dialog telling me the connection was refused or something like that.* »).
- de visualiser et reproduire le problème (par exemple : « *go to www.mlb.com and click on a 'gameday' icon*»).

- de valider l'état de fonctionnement normal du module suite à l'intégration de la nouvelle solution (par exemple : «I downloaded the latest release today (V 1.0.7) installed it and it works exactly the way it did previously with regard to this bug !»).

La description est donc un aspect central de l'organisation de l'activité de résolution au sein de Bugzilla. Elle engendre un transfert important d'informations entre les participants. Les éléments d'informations produits permettent dans un premier temps aux participants, de diagnostiquer la situation rencontrée, et de spécifier par la suite, les tâches qu'ils doivent accomplir pour résoudre le problème.

II.C.2. Définir des rôles

Nous venons de montrer que dans Bugzilla, les tâches et les objectifs à atteindre, ne sont pas spécifiés à l'avance. Ces tâches résultent des échanges importants d'informations concernant l'objet à traiter (le problème et ou la solution à concevoir). De la même manière, les rôles dans Bugzilla ne sont pas clairement établis. En effet, les rôles du rapporteur du bug « *bug reporter* » ou de responsable du bug « *bug assignee* » sont explicitement définis dans les deux premières phases du processus de résolution, nos résultats montrent que les rôles peuvent êtres spécifiés, dans le langage, par des directives qui :

- guide l'exécution de certaines tâches (par exemple : ***make*** *it listen to command rather than click,* **add** *Alt+Down / Up and F4 as equivalents to the dropdown button*);
- attribue une tâche à un contributeur particulier une demande explicite (*par exemple :* ***Mike*** *Bryner Can you take a look at this patch ?,* **Tim** *could you review this and land it on branch and trunk.*);
- verifie que les instructions sont menés correctement (*de The patch can be* ***reviewed*** *I tested it today and didn ' t run into any problems*).

Nous observons un deuxième aspect de l'organisation de l'activité de Bugzilla. Cet aspect, montre que des directives doivent être édictées pour distribuer l'activité, pour combiner les contributions et pour concevoir collectivement la solution au problème.

II.C.3. Expliciter l'activité

L'analyse du langage montre que l'explicitation de l'activité, est l'un des aspects fondamentaux de l'organisation de l'activité, au sein de Bugzilla. En effet, dans grand nombre de leurs messages, les participants déclarent les actions qu'ils ont accomplies. L'action peut être :

- la conception d'une solution (patche) : indiquer le lien qui permet de télécharger la solution (patche) (par exemple : Created an attachment (id=135235) [details]) ;
- la mise à jour d'une solution conçue : indiquer le lien qui permet de télécharger la mise à jour de la solution (par exemple : (From update of attachment 135235 [details])*That checkbox should be controlled by a pref and be integrated with the widget manager mechanism to store it*.) ;
- la vérification les solutions conçues : Une fois une ou plusieurs solutions sont proposées, plusieurs étapes de révisions (par exemple *review=me, super review* =brendan@mozilla.org for trunk checkin), et de test sont mis en œuvre ;
- la fixation de bugs : après vérification, la solution est validée par la fixation du bug (par exemple verified fixed).

II.C.4 Articuler l'activité

Nous avons montré que l'organisation de l'activité au sein de Bugzilla ne repose pas sur les interactions directes entre les participants. En effet, la proportion faible de marques d'émotion et de pronoms personnels à la deuxième personne indique que les interactions dans Bugzilla sont relativement faibles. Dans cette veine, nous avons observé que la majorité des messages postés par les participants, consistent à décrire les situations rencontrées, à donner des directives, et déclarer l'action réalisée, sans obligatoirement s'articuler avec les autres messages.

On remarque cependant que certains rapports de bogues, comportent une part importante de catégories linguistiques, qui expriment que des interactions directes entre les participants ont eu lieu. Le rapport de bogue BR- 362919[67], en est un exemple ; en effet dans leurs messages, les participants interagissent directement et de manière assez fréquente. Ainsi, la communication entre ces participants, se transforme en dialogue entre deux ou trois parties.

Les formes suivantes sont très utilisées :

- des formes qui experiment l'accord ou le dé-accord (*par exemple : Actually, I'm not sure I agree with Gerv here. I understand that RMS agrees Mitchell*);
- des formes qui expriment l'interrogation (par *exemples : Benjamin,could you explain why* ?) ;

[67] Le rapport BR-362919 est l'exemple étudié en introduction de cette section et présenté en Annexe de ce chapitre (deuxième section de ce chapitre- Partie - II.B).

- des formes qui expriment la gratitude (par exemple : *Thanks for writing this, Daniel! I'd say it's a pretty good starting point*.):
- des formes qui expriment l'émotion (par exemple : *Nice one :-) That's what I couldn't work out. Go to it. I'm away for the next nine days; thank you so much for clearing up after me :-) Gerv*).

Nous pouvons remarquer que dans la plupart des messages échangés, les contributeurs désignent nominativement une personne en lui adressant directement un message. Il est intéressant de remarquer que dans ces situations d'interactions directes les participants se connaissent déjà ou essayent de créer des relations personnelles. Cette observation est mérite d'être approfondie dans la mesure ou elle peut nous renseigner sur des modes d'organisations différents. Nous développons cette idée dans la deuxième section du chapitre VI.

L'interaction peu être également observé dans des situations ou le participant, sans contribuer directement à un dialogue, s'articule avec d'autres interventions (messages) en reprenant une partie ou la totalité du message. Les messages de ce type commencent par l'expression « *In replay to comment* ».

Section 3. Conclusions et affinement de la problématique

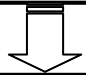
Nous avons montré dans ce chapitre qu'il était possible d'observer à partir de l'analyse du langage, certains aspects de l'organisation de l'activité au sein de Bugzilla, et de définir par la suite les fonctions de la communication dans l'organisation de cette activité.

Dans la section 1, nous avons établi une base méthodologique à partir de laquelle, nous avons pu étudier la communication dans le contexte de Bugzilla. Nous avons commencé la section par la présentation de notre approche: une approche linguistique s'inscrivant dans une démarche purement inductive. Nous avons ensuite abordé l'approche de la linguistique sur corpus qui nous propose une compréhension et une représentation du langage à travers l'identification de genre. Nous avons fini la section en présentons la procédure suivi pour étudier notre échantillon de rapports issus de Bugzilla.

La section 2 a quant elle permis d'identifier à travers l'analyse du langage, les aspects qui caractérisent l'organisation de l'activité, dans le contexte de Bugzilla. Nous apportons ainsi une réponse à notre première question de recherche :

<u>Question de Recherche 1</u> :

Q1 : Quelles sont les caractéristiques du langage utilisé par les contributeurs dans Bugzilla ?



<u>Thèse 1</u> :

TH1*: Le langage utilisé dans Bugzilla présente un genre particulier. Il semble en effet que la* **description** *joue un rôle fondamental dans l'organisation de l'activité, au sein de Bugzilla. Elle intervient principalement au moment où les participants rapportent le problème, pour le rendre visible à l'ensemble de la communauté. La* **description** *est également très présente dans la phase de résolution.*

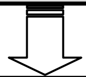
Elle sert d'abord à donner des éléments d'informations sur la méthode adoptée pour concevoir la solution et à confirmer après conception, le bon fonctionnement de la composante, suite à l'intégration de la solution.

Un autre aspect fondamental de l'organisation de l'activité, consiste à déclarer dans le langage les **actions** *qui devraient être faites pour résoudre le problème. L'analyse du langage, montre que ces actions sont* ***dirigées****. Nous avons pu observer que certaines tâches sont guidées par des instructions ou affectées directement à certaines personnes pour les accomplir. Les* **interactions** *directes entre les participants, peu fréquentes dans le langage utilisé, contribuent dans certains rapports de bug dans le but d'articuler la discussion par la création de sorte de dialogue entre les participants.*

A l'issu de ces analyses nous avons identifié les fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla. Nous avons ainsi pu répondre à notre deuxième question de recherche.

<u>Question de Recherche 2</u> :

Q2 : Quelles sont les fonctions de la communication dans l'organisation de l'activité au sein de Bugzilla ?



<u>Thèse 2</u> :

TH2 : *Dans le cadre de Bugzilla, la communication occupe un rôle central dans l'organisation de l'activité de résolution de problème. Nous distinguons quatre de ses fonctions :* ***spécifier les tâches****,* ***définir les rôles****,* ***expliciter l'activité****,* ***articuler l'activité****.*

Nous pouvons dire au terme de ce chapitre, que notre étude exploratoire nous a tout d'abord permis d'identifier les aspects de l'organisation de l'activité au sein de Bugzilla, et puis de proposer quatre fonction de la communication.

Par la suite, cette étude, dans la phase de l'étude de notre première sélection de rapports de bugs, nous a servi à préciser nos choix méthodologiques, concernant notre approche et procédure d'analyse.

Ces choix méthodologiques nous ont permis d'affiner notre problématique et nos questions de recherche, sur l'étude de l'organisation des projets Open Source en rapport avec la résolution de bogues. En effet, en adoptant la méthode de l'analyse de langage, nous pouvons poser un certain nombre de questions :

Premièrement, au cœur de notre recherche et donc de notre problématique, se trouve les questions relatives aux statut l'identité des participants. Nous nous intéressons donc aux questions suivantes :

- ***Q3*** *: Quelle sont les caractéristiques du langage utilisé par les participants ayant des identités différentes dans la communauté ?*
- ***Q4*** *: Ces caractéristiques permettent–elle d'identifier leurs rôle des participants dans l'organisation de l'activité au sein de Bugzilla* ?

Deuxièmement, nous avons démontré dans le troisième chapitre de ce travail que la résolution du problème dépend de l'efficacité de la coordination entre les participants (Malone et Crowston, 1990 ; David et Gosh, 2008 ; Gasser et Sandusky, 2005).

Nous avons également montré que la coordination se déploie à travers la communication entre les acteurs de la communauté reflétée dans le langage utilisée par les acteurs. Nous pouvons ainsi supposer que la variation du langage peut impliquer des processus de coordination différents et par conséquent des trajectoires de résolutions différentes. Une question de recherche est posée:

- ***Q5*** :*Est-ce que la variation du langage utilisé implique des modes de coordination différent?*

CHAPITRE VI-L'ORGANISATION DE LA RESOLUTION DE PROBLEME DANS BUGZILLA : ROLES ET MODES DE COORDINATION

TABLE DES MATIERES

CHAPITRE VI-L'ORGANISATION DE LA RESOLUTION DE PROBLEME DANS BUGZILLA : ROLES ET MODES DE COORDINATION

Dans le chapitre précédent nous avons étudié la communication dans Bugzilla, et identifié plusieurs de ses fonctions à travers l'analyse du langage. Dans ce chapitre, nous spécifions nos analyses en nous intéressons à un nouveau concept : le statut des contributeurs dans la communauté, et en mobilisant une sélection plus importante de rapports de bogues.

En effet, comme nous l'avons souligné dans le second chapitre, de nombreux travaux soulignent le fait que le travail de résolution de problème est distribué de manière asymétrique entre les participants du « cœur » et de la « périphérie » (Von Koch and Schneider 2002; Lee and Cole 2003; Mockus, Fielding and Herbsleb 2002; von Krogh, Spaeth and Lakhani 2003).

Dans Linux Kernel, site principal de l'activité de développement de Linux, lakhani (2005) observent que 75% des massages ont été postés par seulement 11% de contributeurs. Si nous considérons notre échantillon d'étude, nous observons, sur une sélection de 4109 rapports de bogues, liés aux différentes versions de Firefox entre 2001 et 2007, que 87,54% des messages postés sont en provenance de participants du « coeur ».

Ces résultats empiriques soulèvent plusieurs questions sur le rôle des participants dans la résolution de problèmes. Si un nombre limité de participants contribue à l'écriture du code, nous avons besoin de savoir qu'elle est la nature de la contribution des autres participants ? En d'autre terme, si l'écriture du code est le travail des développeurs du « coeur » quelle valeur apporte la contribution de la périphérie dans la résolution de problème ?

La question est encore plus pertinente lorsque nous considérons une période importante impliquant un nombre croissant de contributeurs et favorisant la création dans certains projets, d'échange parallèle, une liste privée qui permet aux développeurs d'échanger en ignorant une partie des contributions (Cox, 1998).

Pour entreprendre l'analyse nous exposerons dans une première section la procédure de l'analyse. Nous présenterons dans une seconde section les résultats de notre analyse.

Section1. Procédure de l'analyse

I.A. Objectif

Comme nous l'avons détaillé dans le second chapitre, les études empiriques identifient plusieurs statuts dans les communautés open source. Selon le modèle de l'oignon (Crowston et Howison, 2005), la structure des communautés Open Source est composée de développeur du centre « *core-developers* », de co-développeurs « *co-developers* », d'utilisateurs actifs « *active users* » et d'utilisateurs passifs « *passive users* ». D'après Gallivan (2001), ces rôles incluent : 1) les chefs de projet qui assument la responsabilité de coordonner le projet, de vérifier et valider les modifications apportées ; 2) les développeurs qui contribuent à l'écriture du code ; et 3) les autres participants qui tout en utilisant le logiciel contribuent à l'identification des bugs, à présenter des rapports de problèmes et tester le code.

D'un autre côté, les résultats empiriques, indiquant qu'un nombre restreint de participants contribuent en majorité dans l'activité d'un projet open source, ont conduit de nombreux chercheurs à faire la distinction entre contributeurs du « coeur » et contributeurs de « la périphéries ». Cette distinction se base sur le droit accordé au contributeur de modifier le code source. Ce droit permet non seulement au contributeur d'avoir une liberté totale d'effectuer des changements mais également de pouvoir soumettre les modifications des contributeurs qui n'ont pas le droit de le faire.

Toutefois, les règles d'obtenir ces droits diffèrent d'une communauté à une autre. Krogh Von et al. (2003) identifient la trajectoire qui permet au contributeur de la périphérie de rejoindre les développeurs du « coeur ». Dans certain projet tel que Gnome, un produit de Lunix, l'accès complet au code source est accordé aux contributeurs qui soumettent des modifications acceptable et à un rythme élevé. Dans d'autres projet, le droit de modifier le code source est octroyé par une procédure formelle, un système de vote, à travers lequel les développeurs du « coeur » accorde l'accès au code source (Mockus et Herbsleb, 2002).

Dans les deux cas, ce qui distingue le « coeur » et la « périphérie » c'est le droit d'accéder et de modifier le code source.

En partant de cette distinction, nous tentons, d'approfondir notre enquête empirique dans le but de répondre aux questions suivantes :

- *Existe t-il une différence entre le langage utilisée par les participants ? spécifiquement entre les participants du « coeur » et de la « périphérie » ?*
- *Si de telle différence existe, est ce quels sont les rôles du « coeur » et de la « périphérie » dans l'organisation de l'activité.*
- *Est-ce que cette différence implique des modes de coordination différent?*

I.B. Méthode de d'analyse

Notre étude est réalisée sur une nouvelle sélection de 4109 rapports de bogues, constitués de 203634 messages. Les messages regroupés forment un corpus que nous appelons « *corp_glob* ». Comme pour la première sélection, ces rapports sont associées aux produits Firefox de Mozilla, et choisis en fonction de leur complexité (les rapports ayant un nombre de patches supérieur à quatre).

Les messages du corpus « *corp_glob* » sont par la suite répartis selon le statut de l'émetteur (contributeur du « coeur » et de la « périphérie »). Suite à cette répartition deux corpus sont formés : un corpus nommé « *corp_N* » constitué par l'ensemble des messages émis par les développeurs du coeur, et un corpus « *corp_O* » qui rassemble les messages émis par les développeurs de la périphérie.
Une fois les deux corpus constitués, nous avons déterminé pour chacun les propriétés du langage utilisé. Ensuite l'analyse des spécificités du langage utilisé nous a permis de tester l'existence de différence significative.

I.B.1. Analyse statistique du langage : analyse des spécificités

Puisque notre objectif consiste à comparer le langage utilisé par les contributeurs du « cœur » et les contributeurs de la « périphérie ». Nous avons choisi l'analyse de spécificité comme méthode.
En effet, l'analyse des spécificités est un outil statistique qui permet de repérer un langage caractéristique d'une partie de texte ou d'un sous-corpus en exhibant tout à la fois les

unités textuelles qu'elle contient en grand nombre (sur représentées) par rapport à la totalité d'un corpus et au contraire, les unités qu'elle contient en très petit nombre (sous-représenté).

Le degré de surreprésentation ou sous représentation statistique des unités est déterminé à partir du calcul d'un indice de spécificité.

L'analyse de la spécificité est issue du modèle hypergéométrique (Lafon, 1984). Elle affecte un indice de spécificité pour chaque unité en fonction de leurs fréquences globale dans la partie et dans tous le corpus. Les valeurs de l'indice sont des entiers. Ces valeurs indiquent le degré de spécificité de la forme, sont signe indique un sur-emploi si elle est positive et un sous-emploi si elle est négative. Lebart et salem 1994.

Dans cette étude, le calcul des spécificités est effectué par le logiciel de textométrie Lexico3 (Salem et al., 2003). Ce logiciel permet d'affecter un degré de spécificité à toute forme lexicale ayant une fréquence supérieure à 10 et un seuil de probabilité égal à 5. Les formes exclusives présentent un indice égale à ***.

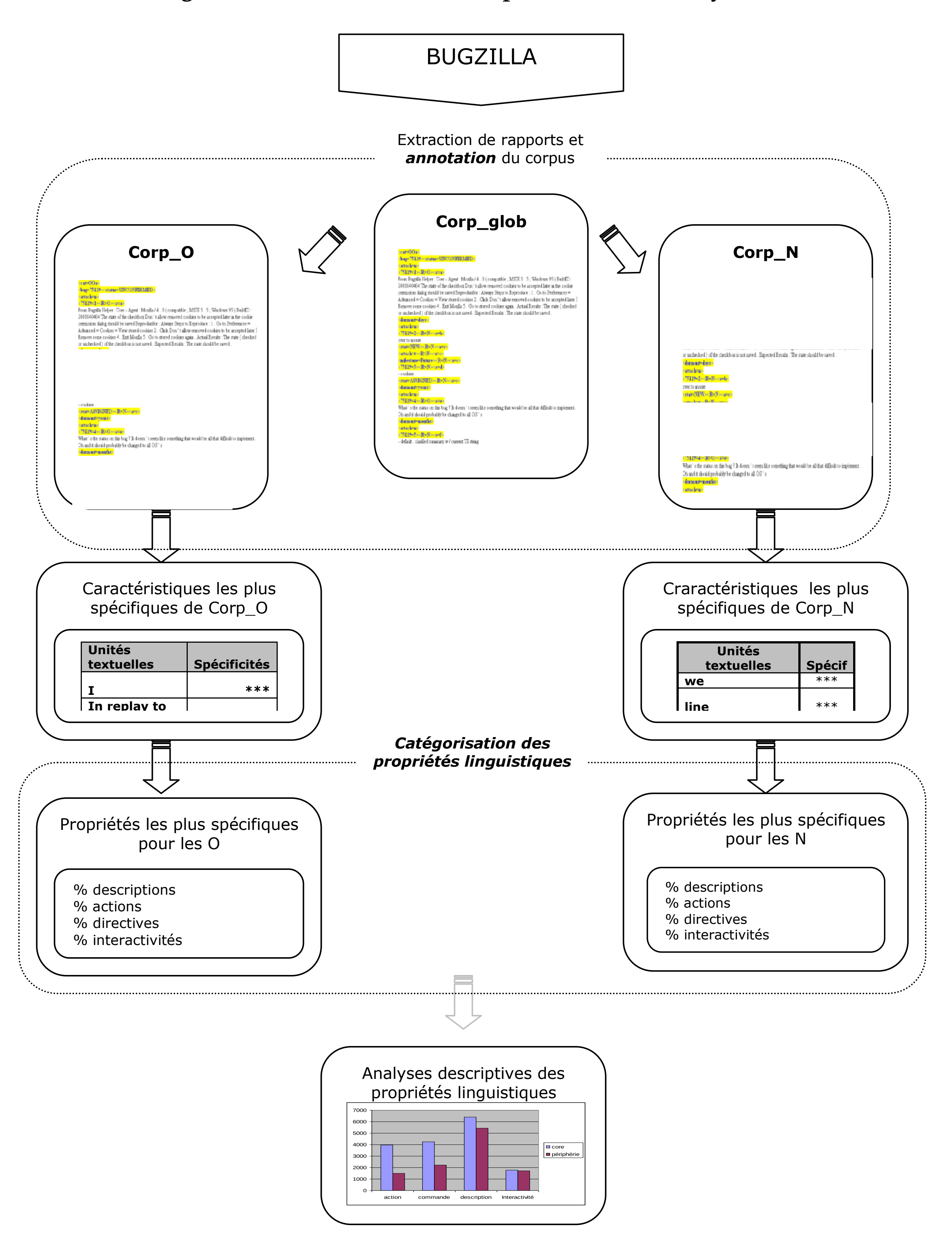
I.B.2 Traitement des données

Après avoir sélectionné notre échantillon d'étude, et choisis la méthode statistique, l'objectif de la phase de prétraitement consiste à passer de données sous leur forme originale à des données exploitables. Une partie importante de cette phase a été détaillé dans le chapitre V. En effet, nous adoptons la même méthode utilisée pour sélectionner et étudier l'échantillon des 692 rapports de bug (cf. chapitre V - I.B), cette fois sur une sélection plus significative de 4109 rapports de bogues.

Comme l'indique la figure 6.1, la présente procédure comporte deux étapes supplémentaires :

- Une étape d'annotation qui permet de découper ou partitionner les corpus à fin de pouvoir distinguer différentes unités d'analyses (« *corp_N* » *et* « *corp_O* »).
- Une étape de catégorisation qui consiste à identifier, dans chaque partie, les propriétés du langage utilisé. Nos premiers résultats (Chapitre V- Section 2) montrent que quarte propriétés en relation avec l'organisation de l'activité de résolution sont identifiée dans Bugzilla : *la description*, *l'action*, *l'interaction* et *les directives*.

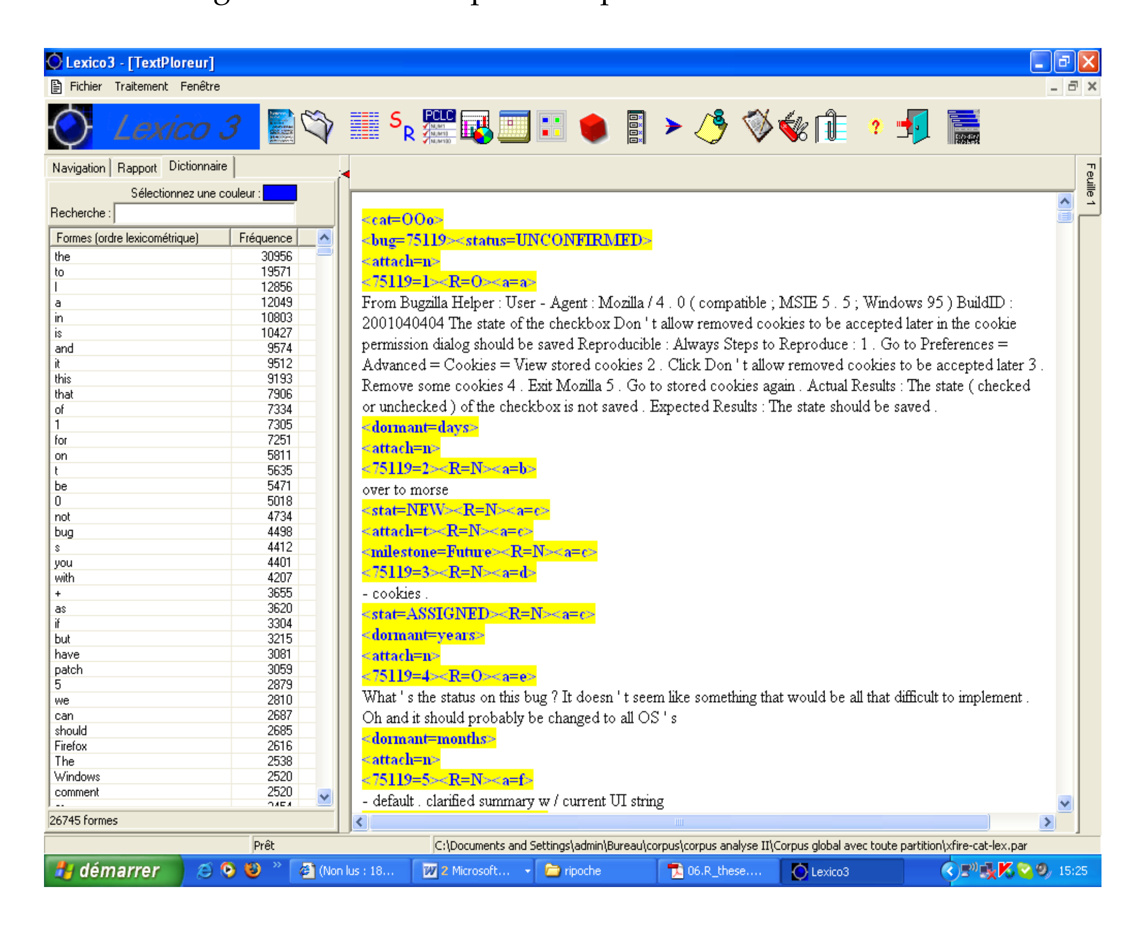
Figure 6.1 : Vue de la deuxième procédure de l'analyse



I.B.2.a Annotation du corpus

Au cours de notre étude, nous cherchons à comparer le langage utilisé par les participants du cœur et de la périphérie. Pour rendre possible ces comparaisons, nous avons introduit dans notre corpus global plusieurs balises[68]. Comme le montre la figure 6.2, le rôle de ces balises est de distinguer différentes parties dans le corpus. Nous avons ainsi introduit une balise **<R= O, U>** pour identifier si le message est émis par contributeurs cœur « O » ou de la périphérie « U ».

Figure 6.2 : Exemples de partition selon Lexico



[68] Selon Salem (2003), une clé ou balise se compose de cinq éléments :< : un chevron ouvrant ; Status : le **type** de la clé ; = : le signe "égal" ; O : le contenu de la clé et > un chevron fermant. Exemple : < Status=O>. Il montre que l'insertion de clés constitue une phase importante dans la préparation du texte. Les clés introduites permettront ensuite à l'utilisateur d'effectuer des comparaisons à partir des parties du corpus qu'elles découpent.

I.B.2.b Sélection des unités textuelles et identification des catégories linguistiques

Comme l'indique le tableau 6.1, lorsque l'on prend en compte les différentes formes et segments de formes identifiées dans chaque corpus, le nombre d'unités augmente de façon importante.

Pour réduire le nombre d'unités textuelles à traiter, nous avons choisi les unités qui présentent un degré de spécificité positif. Nous limitons ainsi nos unités textuelles à 1912 formes pour la partie N et 1620 formes pour la partie O.

Une fois les unités textuelles générées, nous les avons soumis à l'outil d'analyse Sphinx dans le but de dégager une répartition des catégories linguistiques. Nous rappelons que le même outil à été utilisé, pour le même objectif, dans la première étape de l'analyse (cf. chapitre V- I.B.3).

Tableau 6.1 : Principales caractéristiques lexico métriques du corpus global et ses parties

	Nombre de formes	Nombre d'unités (occurrences)
Corp_glob	59025	1946117
Corp_N	49862	1322515
Corp_O	18604	17177

I.B.2.c Catégorisation des unités linguistiques

Selon Miles et Huberman (2008), la catégorisation « permets au chercheur de *trouver*, et *rassembler* rapidement les *segments* relatifs à une *question de recherche spécifique*» (Miles et Huberman, 2008, p.). Notre objectif ici est d'attribuer aux formes ou aux segments linguistiques, identifiés dans chaque partie, des propriétés en relation avec l'organisation de l'activité au sein de Bugzilla.

Bien que la l'identification des catégories linguistiques soit réalisée automatiquement, la catégorisation est effectuée manuellement en suivant la démarche décrite par Bardin (2005) :

« *La catégorisation se réalise par une opération de classification d'éléments constitutifs d'un ensemble par différenciation puis regroupement par genre (analogie) d'après des critères préalablement définis*.» (Bardin, 2005, p.) :

Nous expliquons ce choix par deux raisons :

D'abord une catégorisation automatique s'avère peu efficace dans un environnement tels que Bugzilla. Selon Ripoche (2006), l'application d'un correcteur orthographique ou l'utilisation d'un analyseur syntaxique dans le cadre d'une annotation automatique indiquent des taux d'erreurs trop élevées. En effet, sur un échantillon de messages tirées de Bugzilla, Ripoche (2006) montre que « *moins d'un tiers de ces messages sont correctement analysées par un parseur syntaxique superficiel, alors que ce même parseur obtient des scores au-delà des 90% sur des corpus plus conventionnels (Munoz et al., 1999)* ». Il indique également que « *l'application d'un correcteur orthographique est rendue inefficace par l'existence d'un grand nombre de mots inconnus qui sont souvent faussement convertis en terme de lexique*.» Ripoche (2006).

En plus, la segmentation automatique du corpus textuel en mots, formes ou segment simplifie la tâche de catégorisation manuelle en permettant, de se concentrer sur la catégorisation des unités les plus significative (coefficient de spécificité positif). D'autre part, en ayant, dans une première phase de l'analyse (cf. chapitre V), un contact direct avec les données et une vision plus profonde du phénomène, une catégorisation manuelle semble plus fiable. Cette vision s'approche de celle de Strauss et Corbin (1990), selon laquelle l'analyste développe ses catégories au contact avec les données qu'il étudie et en fonction des phénomènes qu'il observe dans ces données.

Section 2. Résultats de recherches : analyses et conclusions

La section précédente a permis de détailler notre cadre et procédure d'analyse. Dans cette section, nous mettons en avant les résultats de ces analyses. Trois types de résultats sont présentés et analysés :

- Les premiers résultats proposent une première vue sur la contribution des participants du coeur et de la périphérie à l'activité de Bugzilla (II.A).
- Les seconds résultats portent sur la variation de ces contributions sur la base de l'analyse du langage, les rôles des contributeurs dans l'organisation de l'activité sont par la suite identifiés (II.B).
- Enfin, les troisièmes résultats distinguent sur la base de ces variations trois modes de coordinations l'évolution de cette variation selon le statut du problème (II.C).

II.A Première vue sur la contribution du « cœur » et de la « périphérie » à la résolution de problèmes

Si nous considérons qu'une activité de résolution est identifiée par des actions comme, concevoir différents patches, ou bien assurer la sélections des propositions des participants (Markus et al., 2000 ; Bird et al., 2007 ; Von grogh, 2003), nous constatons en se basant sur les illustrations présentées en figure 6.3, 6.4, que la contribution des participants du cœur est globalement plus significative que la celle de contribution de la périphérie.
En effet, la figure 6. 3 illustre la distribution des fréquences des messages émis par les participants du cœur et de la périphérie selon le numéro version du patch[69]. Nous pouvons remarquer qu'en majorité, les patchs sont conçus par les participants du cœur.

Nous observons également les mêmes tendances lorsque nous comparons la distribution des messages émis par les participants du cœur et de la périphérie en fonction du statut du bogue (cf. figure 6.4).

Ces observations semblent concorder avec les résultats identifiés dans la littérature qui montrent que, pour la majorité des projets open source, les actions les plus critiques (la

[69] Nous rappelons que pour résoudre un problème plusieurs versions de patchs peuvent être conçues. Pour chaque version nous attribuons un numéro, la figure 6. 3 présente la distribution de 30 versions de patch.

conception de patchs, la validation de solutions, etc.) sont réalisées par les contributeurs du cœur et les administrateurs du projet (Lee et Cole, 2003 ; Ghosh et David, 2003). Les tâches les plus simples comme la déclaration de problème, le teste des solution, etc., sont quand à elles effectués par les contributeurs de la « périphérie » (Mockus et al.; 2002). Cette dernière observation est confirmée dans le cadre de notre étude puisque 67,58 % des messages qui concernent la déclaration d'un problème sont en provenance de la périphérie et 56,45% des participants de la périphérie déclarent une seule fois un problème.

Figure 6.3 : Distribution des fréquences de patchs émis par version pour participants de cœur « N » et de la périphérie « O »

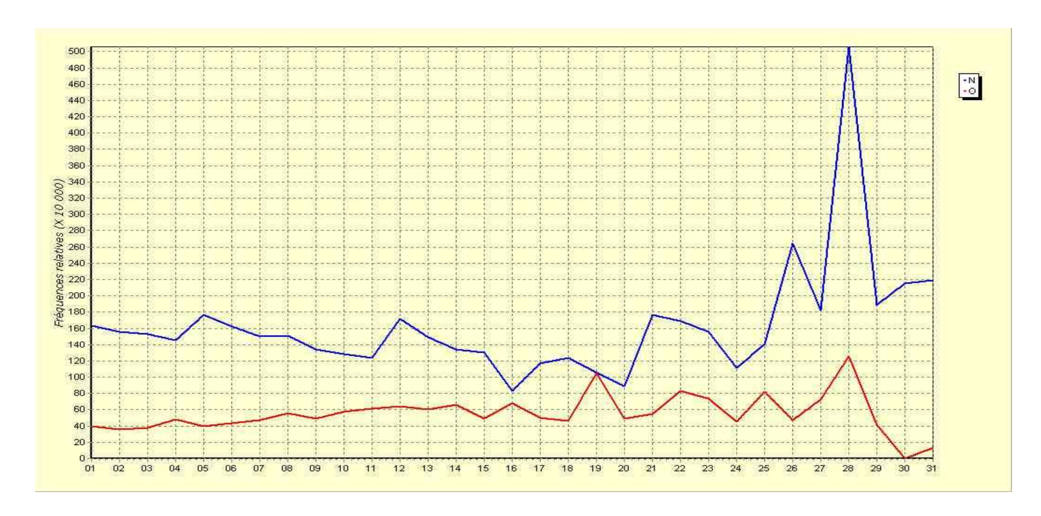
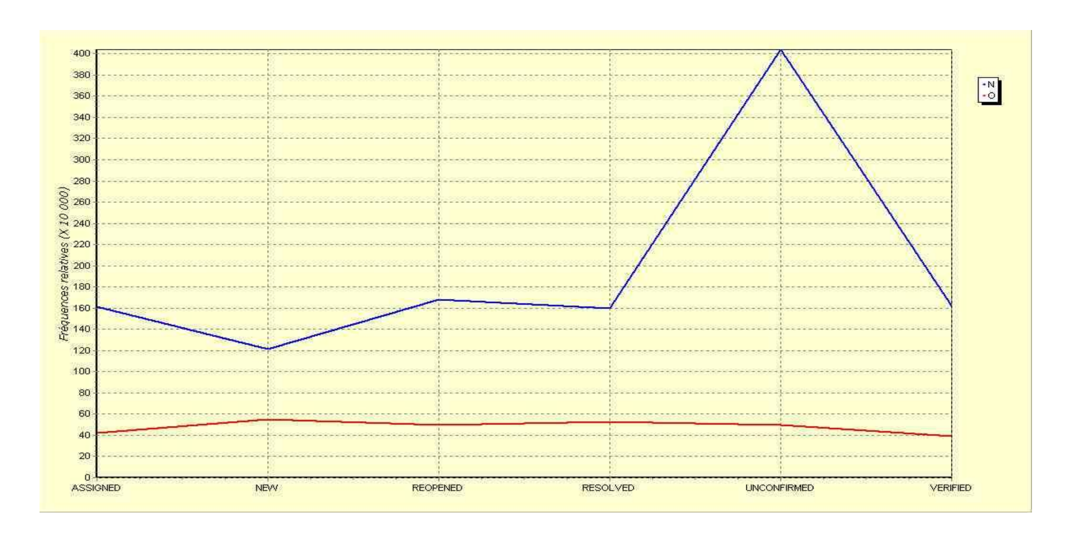


Figure 6.4 : Distributions des fréquences de messages émis par statut du bug pour les participants de cœur « N » et de la périphérie « O »



A ce niveau d'observation, nous remarquons que le travail technique (conception de solution, fixation de problème, etc .) est la charge des participants du cœur et que la contribution de la périphérie n'est pas significative à ce niveau.

Cependant, si le travail de résolution est pour la plus du temps technique et n'imposent pas, normalement, des échanges fréquents, les développeurs du cœur multiplient, pourtant, les échanges avec les autres contributeurs. En effet, 61,73% de messages postés dans les 4109 rapports sélectionnés en provenance des contributeurs du cœur ne contiennent pas de patches.

Ces constatations montrent le besoin des participants du cœur de communiquer avec la périphérie et renforce la thèse selon laquelle la contribution de la périphérie et aussi importante que celle du cœur dans l'activité de résolution (Lakhani, 2006). Pour mieux caractériser ces contributions nous tentons d'étudier, dans la seconde partie de ce chapitre, les échanges dans leur forme d'origine à travers une analyse du langage.

II.B Rôles du cœur et la périphérie dans l'organisation de l'activité au sein de Bugzilla

Dans le présent développement nous nous penchons de nouveaux sur l'étude de langage. Notre objectif est de comparer la contribution du cœur et de la périphérie dans l'organisation de l'activité de résolution de problème au sein de Bugzilla.

L'activité ici n'est pas représentée comme une série d'actions figées mais comme une structure en relation avec la nature de la communication et du langage échangé. En d'autre terme en analysant l'activité de résolution nous nous concentrons sur ce que ces participants communiquent.

Nous rejoignons ainsi l'approche développée par Winograd (1988) qui considère la communication comme une structure centrale qui permet d'anticiper et de coordonner les actions :

> « *Conversations for action are the central coordination structure for human organisations. We work together by making commitments so that we can successfully anticipate the actions of others and coordinate them with our own.* » (Winograd, 1988, p.10)

II.B.1 Caractérisation du langage du coeur

II.B.1.a Spécificités linguistiques et description de la contribution du cœur

Pour mieux comprendre la nature de la contribution des participants du cœur, nous étudions dans ce qui suit les caractéristiques linguistiques des unités textuelles les plus spécifiques.

Nous rappelons que cette analyse est basée sur le corpus *Corp_N* . Comme nous l'avons expliqué dans la première section de ce chapitre, ce corpus est constitué de l'ensemble des messages postés par des développeurs du « coeur» dans une sélection de 4109 rapports de bug. Les unités textuelles qui forment ces messages sont réduites, en raison de leurs nombres importants, aux unités ayant un coefficient de spécificité supérieur à 10.

Les résultats d'analyse des spécificités, présenté dans le tableau 6.3, indiquent que la forme la plus spécifique (coefficient de spécificité présenté par ***) au langage du cœur est le pronom personnel de première personne du pluriel « *we* ».

En effet, le pronom personnel *« we »* est souvent utilisé par les contributeurs du cœur dans le but d'exprimer un avis ou un besoin commun *« we need »* (coef de spécificité=16) *; « we should »* (coef de spécificité=15).

Le pronom personnel de troisième personne est également souvent employé par les contributeurs du cœur (coef de spécificité= 22). Comme énoncé, son usage permet à sont émetteur de donner des instructions concernant une situation donnée, dans le but d'exprimer l'obligation, le besoin ou la possibilité. Biber et Conrad (2009) soulignent également la fréquence de l'usage des pronoms personnels de troisième personne par les « experts » des forums électroniques, en comparaison aux « novices ».

De plus, l'usage fréquent du pronom « *we* » montre que les participants du cœur tentent d'inclure les autres participants dans la discussion, en s'identifiant en tant que membre appartenant à la communauté. Ces constations reflètent donc la notion de communauté et de construction collective tels nous l'avons présenté dans le second chapitre.

D'une manière générale, les noms sont très utilisés au sein de Bugzilla. Les résultats montrent que les formes nominales les plus spécifiques au langage du cœur sont : « a=asa »70, « rv= me», « rdf », « NS », « line » et « nc ».

[70] Comme pour le cas de la majorité des forums électroniques, le langage technique utilisé au sein de bugzilla est souvent abrégé. En effet, la lettre « a »est l'abréviation du nom « assignee » ; rv est l'abréviation du nom

Comme nous pouvons le constater, ces noms sont pour la plupart, des noms techniques abrégés. D'après Liénard (2005), les procédés de simplification d'écriture, comme l'abréviation sont spécifiques au genre des forums électroniques.

Dans Bugzilla, ces noms sont destinés à décrire un phénomène rencontré à travers l'adoption de modèles graphiques (*rdf*), de systèmes de simulation informatique (*Ns*), ou d'utilitaires qui permet d'identifier des programmes informatiques (*nc*).

Selon Ripoche (2006), leurs usages facilitent la collaboration entre les participants par la création de langage ou de répertoire commun. Ce langage permet, en plus de ses propriétés descriptives, de spécifier certaine tâche comme par exemple la révision du patch (« *rv=mPrefs->GetBoolPrefrv=tim* », « *Need the sr= now etc.* ») ou l'assignation d'un bug à un développeur (*« a=asa »*).

Il ressort de ces résultats que les verbes présentent des cœfficients de spécificités moins importants par rapport aux noms et aux pronoms personnels. Le verbe est cependant présent avec une densité lexicale[71] importante. Nous comptons sur 1871 unités textuelles, ayant une spécificité positive, 30% de verbes et 21% de noms.

Nous avons également constaté que ces verbes sont souvent utilisés pour indiquer que des actions sont effectuées comme par exemple la création d'attachement (*#CATTACH*, spécificité=***) ou la mise à jour d'un attachement (*#UATTACH*, spécificité=49), mais aussi pour donner des instructions ou demander une aide explicitement pour la réalisation d'une tâche.

« revision » ; « rdf » est l'abréviation de « Ressource Description Framework » ; « Ns » est l'abréviation de « Network Similator » et « nc » est l'abréviation de « Net Cat ».

[71] Une densité lexicale est calculée à partir nombre des différentes formes d'une catégorie.

Tableau 6.3 Présentation des unités textuelles les plus spécifiques au langage utilisé par le cœur[72]

Unités textuelles	Spécif
we	***
line	***
#CATTACH	***
rv	***
content	***
rdf	***
NS	***
nc	***
components	***
#UATTACH	49
Help	40
patch	39
branch	39
Changes	24
need	22
it	22
attach	19
check	18
builds	18
looks	17
unsigned	17
verified	17
moving	17
we don t	16
we should	15
bugs	15
for	13
browser	13
Fixed	11
doing	10

II.B.1.b Le rôle du cœur dans l'organisation de l'activité au sein de Bugzilla

Les figures 6.3 illustre la distribution des catégories linguistiques identifiées dans le chapitre précédent, à savoir les catégories description, actions, directives et interactivité. Comme expliqué, cette distribution est élaborée sur la base d'une catégorisation manuelle de notre échantillon *Corps_N*.

Les résultats de cette analyse indiquent que le langage utilisé par les participants du cœur est essentiellement descriptif (42,45%). Si nous considérons le langage de point de vue de

72 Pour une raison de lisibilité, seulement 30 unités textuelles qui ont fait l'objet d'interprétation, les autres unités sont présenté dans l'annexe 6.1.

l’action, nous constatons, que les participants du cœur utilisent souvent les unités linguistiques pour expliciter une action ou pour anticiper les actions à réaliser (26,30%) avec des proportions équivalentes de directives (28,19%). Finalement on note une faible proportion (3%) d’unités linguistiques indiquant la présence d’interactions directes entre des participants du cœur et les autres contributeurs.

La figure 6.3 montre également que les proportions d’actions, de descriptions, et de directives sont encore plus considérables après la conception des premiers patches. Ces constations confirment nos premières observations (II.B), indiquant que le cœur intervient non seulement dans les phases critiques de conception mais aussi dans les phases de révision, de validation et d’intégration de solutions, réalisés dans les dernières phases de résolution. Ces constatations peuvent s’expliquer par le fait que le projet Mozilla est fortement structuré et que l’intégration des contributeurs n’est pas complètement libre et nécessite qu’elle soit validée par des administrateurs, désignés par la communauté.

Figure 6.3: Distribution des catégories du langage utilisées par les participants du cœur

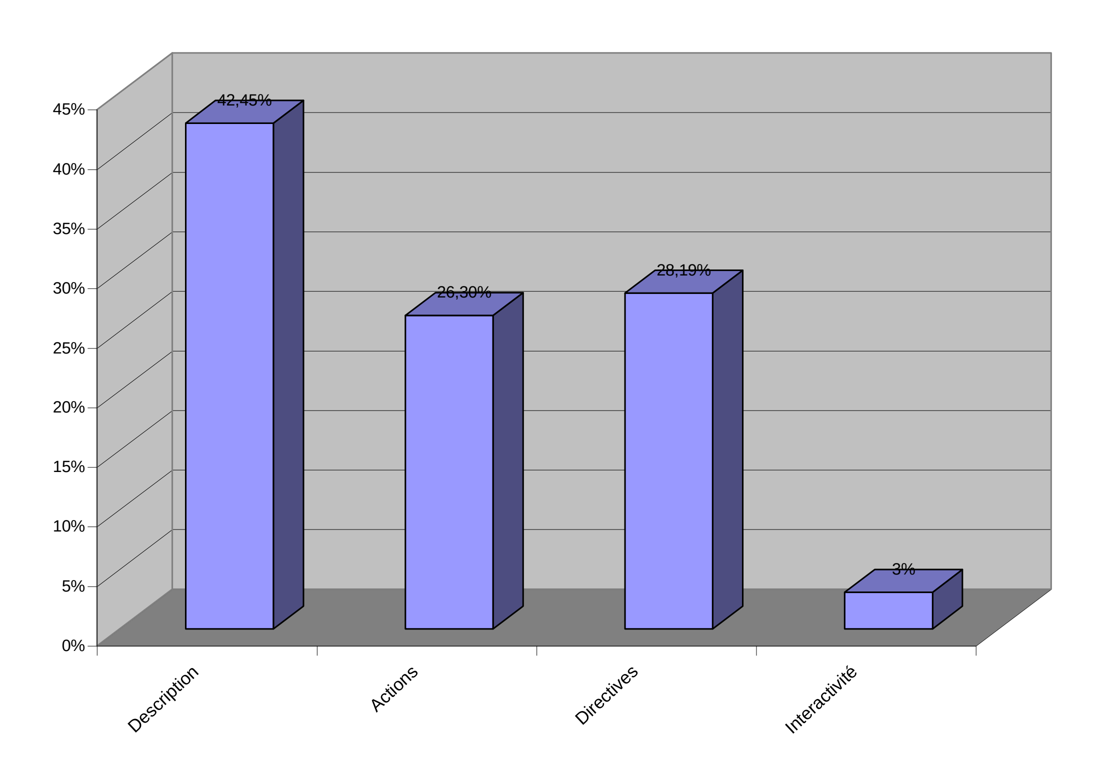
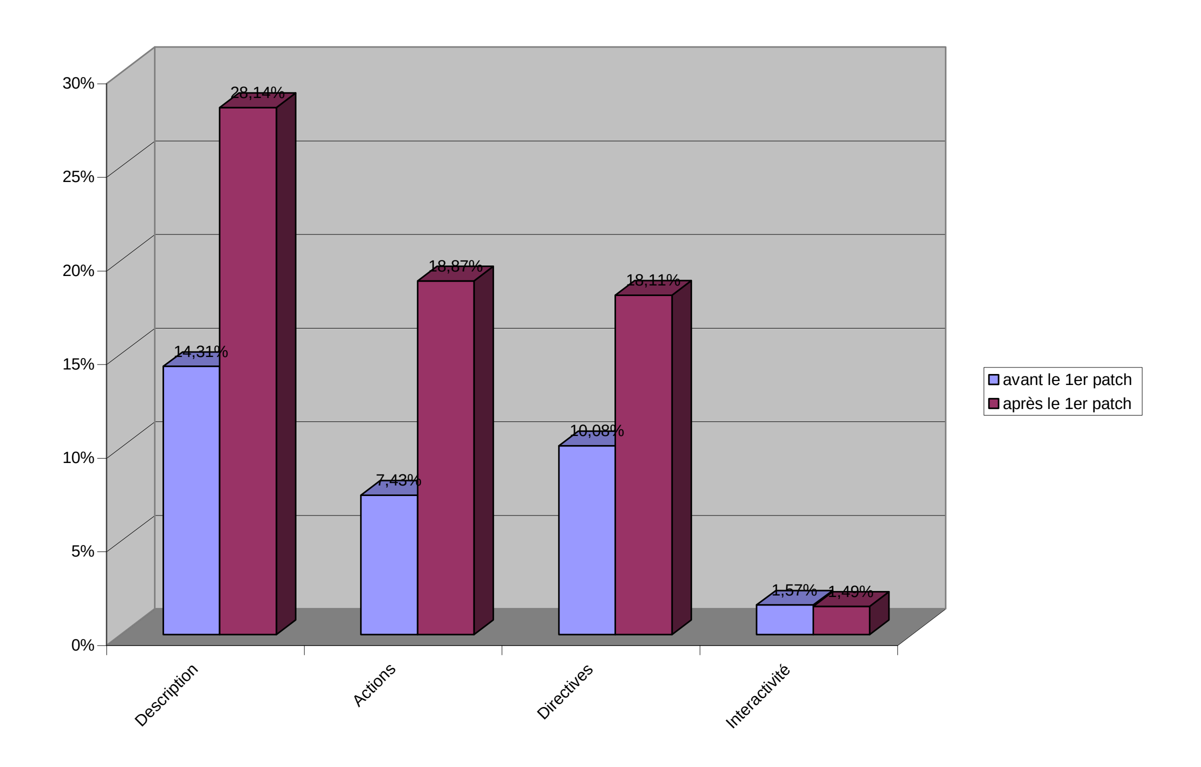


Figure 6.4: Distribution des catégories du langage utilisées par les participants du cœur avant et après le premier patche



A ce niveau d'observation, nous pouvons dire que le rôle des participants du cœur dans la coordination et plus généralement dans l'organisation de l'activité de résolution au sein de Bugzilla consiste à :

- *Décrire le contexte*

D'une manière générale, ce qui caractérise les communautés médiatisées tels que Bugzilla sont les échanges intenses en informations. Les proportions importantes des unités linguistiques de nature descriptives montrent qu'une grande partie des aspects informatifs ont trait à la description du problème. Il est en effet important que les développeurs du « coeur» mettent à la disposition des contributeurs l'ensemble des informations concernant le contexte du problème et surtout le contexte qui permet de concevoir collectivement une solution. La coordination de l'activité passe donc avant tout par la description.

- *Énoncer l'action*

La coordination se fait par l'intermédiaire de la déclaration d'action. Comme nous l'avons détaillé précédemment, 26, 30% des unités linguistiques utilisés par les participants du

cœur déclarent la réalisation d'une action, nous avons également observé que ces proportions sont beaucoup plus considérables après la conception du premier patche. Nous pouvons donc constater que dans un contexte ou l'activité consiste en plusieurs tâches, dans lequel les rôles ne sont pas définis, déclarer dans le langage que des actions sont effectuées constitue un moyen qui permet aux participants du cœur de coordonner l'activité.

- *Diriger l'action*

On remarque que dans le langage adopté par les participants du cœur, un taux d'instructions plus important après l'émission du premier patche (cf. figure 6.4). Les participants du cœur coordonnent ainsi l'activité en guidant les autres contributeurs dans leurs conceptions. Nous avons également observé qu'une bonne proportion de verbes utilisés concerne une demande explicite d'action. Nous pouvons en déduire que l'activité de résolution de problème est orchestrée par les directives des participants du cœur. De plus, les proportions importantes d'unités linguistiques de nature descriptive montrent que ces directives nécessitent d'importants échanges d'informations, qu'il faut également coordonner.

- *Légitimer l'action*

Les observations faites sur les proportions d'interactions directes qui ont lieu entre les participants du cœur montrent que la coordination ne se fait pas par l'intermédiaire des interactions directes entre des acteurs. Ces constatations rejoignent la caractérisation faite par Ripoche (2006) suggérant que « *l'interaction dans Bugzilla est beaucoup moins directes et interactive que dans les dialogues plus conventionnels* » (Ripoche, 2006, p 140). En effet, Ripoche (2006) présente les interactions au sein de Bugzilla comme des interactions de type *black-board* (Hayes-Roth, 2006)[73] où les participants coopèrent à réaliser une tâche (la résolution de problèmes) sans qu'il ait des interactions directes entre les participants.

Nous notons également que l'usage fréquent de pronoms personnel de première personne au pluriel par les participants du cœur montre que même si ces derniers n'interagissent pas directement avec la communauté ils ont toutefois besoin de s'identifier. Ceci peut

[73] Selon Ripoche « La notion black board (Hayes-Roth, 1979) provident de la recherche sur les systèmes multi-agents. Dans un système de ce typen les particiapnts apportent des informations dans un espace commun (le black bord) qui sont ensuite synthétisées au cours d'un processus de raisonnement centralisé.» (Ripoche, 2006, p 140)

s'expliquer par le fait que dans Bugzilla les rôles ne sont pas clairement établis, s'identifier à un groupe ou à la communauté en générale, permet au participants du cœur (administrateurs) d'acquérir une certaines légitimité dans le but de contrôler la résolution du problème (O'Mahony , 2003).

Cette observation est confirmée dans les travaux Markus et al. (2009), d'après lesquels la collaboration dans le cadre des projets Open Source nécessite que les procédures de contrôle soient légitimées par les niveaux les plus bas de la pyramide :

> « *OSS collaboration implies a very peculiar nature of control, which can be centralized into a leader (as happens for Linux, Raymond, 1998; Lerner and Tirole, 2002) but needs to be continually legitimated by those constituting the "lower layers" of the pyramid.* » (Markus et al, 2009, p 8)

II.B.2 Caractérisation du langage de la périphérie

Nous venons d'identifier plusieurs propriétés déterminant le rôle du coeur dans l'organisation de la résolution de problème au sein de Bugzilla. Dans cette partie, nous adoptons la même méthode pour caractériser le rôle de la périphérie.

La présente analyse est basée sur le corpus *Corp_O*, regroupant l'ensemble des messages émis par les participants de la périphérie dans notre sélection des 4109 rapports de Bugs. Nous suivons la même démarche de sélection et de catégorisation décrite dans la partie précédente, nous avons ainsi retenu les unités ayant un coefficient de spécificité supérieur à 10.

II.B.2.a Spécificités linguistiques et description de la contribution de la périphérie

Les résultats présentés dans le tableau 6.4, montre que la forme la plus spécifique au langage utilisé par les contributeurs du cœur est le pronom personnel « ***I*** » (coef de spécificité = ***). En utilisant le pronom personnel « ***I*** », les contributeurs de la périphérie s'investissent dans ce qu'ils énoncent comme informations, sans s'associer à d'autres participants. En effet, l'usage fréquents de segments formes telles que « ***I have, I see, I think, I use, I agree, I dont have, I'm sure*** », montre que la périphérie interviens dans la résolution pour exprimer une pensée, suggérer, ou produire des informations concernent l'environnement du problème.

Nous observons également que le verbe « ***replay*** » est très fréquent dans le langage utilisé par la périphérie (coéf de spécificité= ***). Il est souvent cité dans la phrase « ***In replay to comment*** », ayant pour fonction la reprise de segment du commentaire auquel on répond pour

l'évaluer ou le compléter. D'après Marcoccia (2004), cette forme permet la mise en scène de la dimension interactionnelle des échanges en les rendant explicites. Les contributeurs de la périphérie tentent ainsi non seulement de créer des liens entre les messages, et recontextualiser leurs interventions, mais également de développer des relations avec les autres contributeurs dans l'objectif de prouver leur intérêt au projet, et d'intégrer la communauté.

Les autres verbes signalés dans le langage de la périphérie, présentant des indices de spécificité significatifs (supérieur à 10), sont souvent utilisés pour décrire une situation en rapport avec le problème rencontré. Nous notons par exemple une forte utilisation des verbes avoir et être au présent (« ***have*** » (coef de spécificité= 32), « ***are*** » (coef de spécificité= 19).

Les noms comme « *Mozilla », « mail», « firefox », « file* », « *problem* » sont présent avec des coefficients de spécificités important (coef de spécificité= (***)) pour les trois premières formes de noms à 36 pour file, et à 23 pour « *problem* »). Comme pour les verbes, les noms servent à désigner le problème en se référant à des emplacements (liens, adresses, etc.) permettant aux autre participants de visualiser le problème en le reproduisant.

Associées aux noms, les prépositions sont également utilisées par la périphérie (« *in* » (coef de spécificité=22), « *by*» (coef de spécificité=13), «*on*» (coef de spécificité=12), etc.). Comme expliqué, elles servent à attribuer des propriétés contextuelles. Leur rôle est donc principalement descriptif.

Nous observons également que l'adjectif « *duplicate* », très utilisé de manière générale dans Bugzilla, est particulièrement représenté dans le langage de la périphérie (coef de spécificité= (***)). Fréquemment cité dans la phrase «*Bug 84066 has been marked as a duplicate of this bug* », en utilisant cet adjectif, les contributeurs de la périphérie participent à la résolution en fournissant des informations permettant de reproduite le problème ou de l'identifier à un bug similaire (bug duplicate), dans le but d'orienter la résolution du problème.

Tableau 6.4 Présentation des unités textuelles les plus spécifiques au langage utilisé par la périphérie[74]

Unités textuelles	Spécificités
I	***
In replay to comment	***
Mozilla	***
mail	***
Firefox	***
duplicate	***
file	36
Steps to Reproduce	36
have	32
my	23
problem	23
click	22
in	22
option	20
are	19
Actual	18
confirm	16
tried	16
open	14
this bug	14
user	14
by	13
and	13
right click	12
for me	12
the	12
on	12
still	10
of	10
will	10

II.B.1.b Le rôle de la périphérie dans l'organisation de l'activité au sein de Bugzilla

Nous donnons dans la figure 6.5 la distribution des quatre catégories linguistiques pour le corpus *Corp_O*. Comme pour l'analyse précédente, cette distribution a été obtenue en convertissant le corpus dans les quatre catégories identifiées : description, actions, directives et interactivité.

A partir de ce résultat, nous constatons un taux important de description (49,39%) en comparaison avec les autres catégories. Nous observons par ailleurs des taux moins

[74] Pour une raison de lisibilité, seulement 30 unités textuelles qui ont fait l'objet d'interprétation, les autres unités sont présenté dans l'annexe 6.2.

importants des formes d’interactions (18,47%) et des formes rapportant qu’une action est effectuée (15,68%). On note finalement de faibles proportions (8,91%) de directives dans le langage utilisé par les participants du cœur.

Nous pouvons remarquer que ces résultats rejoignent les observations faites dans la littérature empirique, notamment les travaux de Lakhani et al. (2006); Crowston et Howison (2003). En effet, ces travaux postulant que la contribution de la périphérie dans les communautés Open Source consiste principalement à déclarer des problèmes, à demander des instructions, ou de consignes le concernant sans intervenir de manière significative à sa résolution (proposition de patchs). Ceci explique donc les fréquences élevées des messages de la catégorie description suivit de la catégorie interactivité. Nous rappelons ici que la dimension interactionnelle est représentée par des formes, comme par exemple « *replay* », dans le but de créer des liens entre les messages ou de les recontextualiser. L’interaction n’est donc pas directe et ne reflète pas une structure de dialogue.

La distribution présentée dans la figure 6.5 confirment nos observations étant donné que les fréquences des catégories descriptions, actions, et directives sont plus important avant la conception du premier patche, c'est-à-dire avant de commencer de concevoir une solution au problèmes.

La contribution de la périphérie est donc plus significative dans les premières phases de la résolution, phases qui se caractérisent par un très fort taux de description et donc d’échanges d’informations[75] qu’ils faut également coordonner.

Les constatations faites sur les distributions des catégories linguistiques et sur les spécificités du langage utilisé par les participants de la périphérie nous amènent à identifier leurs rôles dans l’organisation de la résolution de problème au sein de Bugzilla. Nous distinguons dans ce qui suit trois rôles.

---

[75] La figure 6.2 présentée dans la première partie de cette section (II.A), indique que le nombre de message est plus important lorsque le statut du bogue est Unconfirmed. Nous qui correspond à la première phase du processus de résolution de problème.

Figure 6.5 : Distribution des catégories du langage utilisées par les participants de la périphérie

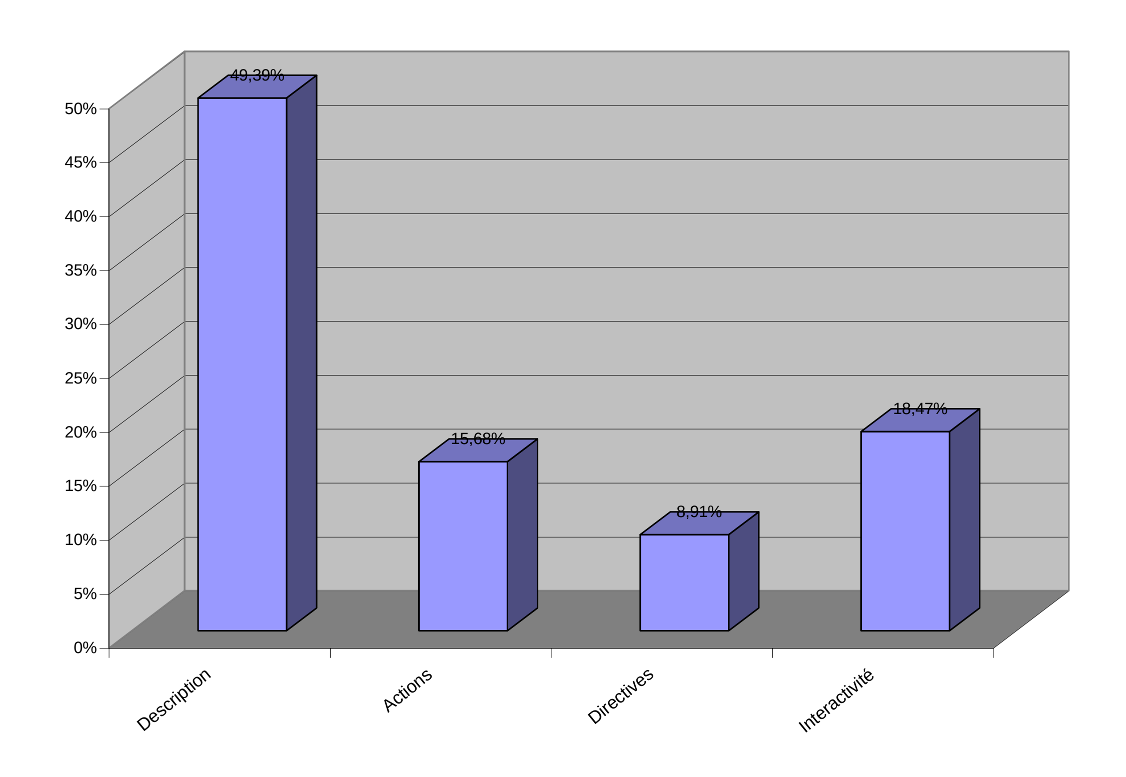
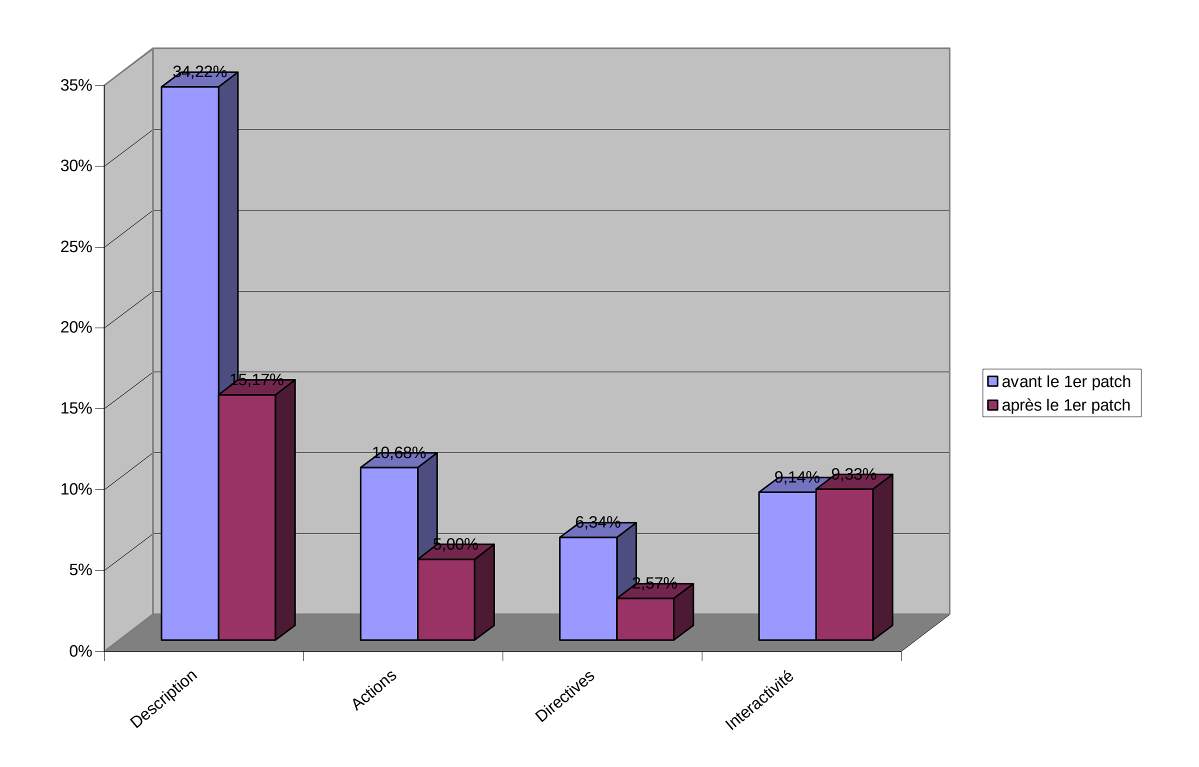


Figure 6.6 : Distribution des catégories du langage utilisées par les participants de la périphérie avant et après le premier patche



- *Attribuer des propriétés contextuelles* :

Premièrement, nous avons pu observer le rôle central de la périphérie dans l'attribution des propriétés contextuelles aux problèmes. En effet, nous avons montré que l'usage des verbes, des noms propres et des prépositions avaient pour fonction de décrire des situations rencontrées, ou à de se référer à des emplacements pour décrire les situations rencontrées. Ces observations ont été conformé par le taux considérable de description dans le langage utilisé par la périphérie.

- *Articuler les interventions* :

Ensuite, les observations faites sur la nature des interactions dans Bugzilla montrent que les marques d'interactivités (échanges directs) sont moins fréquentes dans le langage utilisé par les participants de la périphérie (18, 47%). Nous avons également constaté que l'interaction dans Bugzilla n'est pas structurée autour d'un dialogue direct entre les participants. Néanmoins les résultats sur les spécificités du langage utilisé par la périphérie montrent que ces participants contribuent à la coordination l'activité de résolution en articulant les interventions. En effet, dans leurs messages la périphérie tente de créer des liens entre les messages sans avoir à échanger ou à dialoguer directement avec les autres participants. Ces liens permettent alors de recontextualiser les interventions en particulier dans les phases de définitions de problèmes, se caractérisant généralement par un échange important d'informations entre participants qu'il faut souvent articuler.

- *Orienter la résolution*

Enfin, les proportions faibles de directives montre que la périphérie n'intervient pas de manière directe pour diriger l'activité de résolution. C'est en effet le rôle des participants du cœur ou plus spécifiquement des administrateurs d'attribuer les tâches, de vérifier, et de valider l'intégration des contributions. Néanmoins, l'usage de certaines formes linguistiques comme par exemple « *duplicates* » montre que la périphérie oriente indirectement la résolution en déclarant le bogue comme étant identique à un autre. De plus, l'identification d'anomalies par la périphérie, généralement liée à l'intégration d'une nouvelle solution oriente également la résolution vers la recherche de nouvelles solutions parfois même après la fixation du bogue impliquant la réouverture du bogue.

II.C. Synthèse et comparaison

Les résultats présentés dans les sections précédentes définissent les catégories et les spécificités des échanges pour les développeurs du core et de la périphérie. Dans cette partie, nous présentons une synthèse ainsi qu'une comparaison des deux analyses.

La figure 6.9 montre que les commentaires de catégorie action et commande proviennent en majorité des développeurs du « core ». Nous constatons des variations moins importantes entre « core » et « périphérie » pour les catégories description et interactivité.

Globalement, bien que leurs spécificités différent en fonction de l'identité de l'émetteur (« Insider » ou « outsider »), les messages de catégorie description sont les plus fréquents dans les deux corpus.

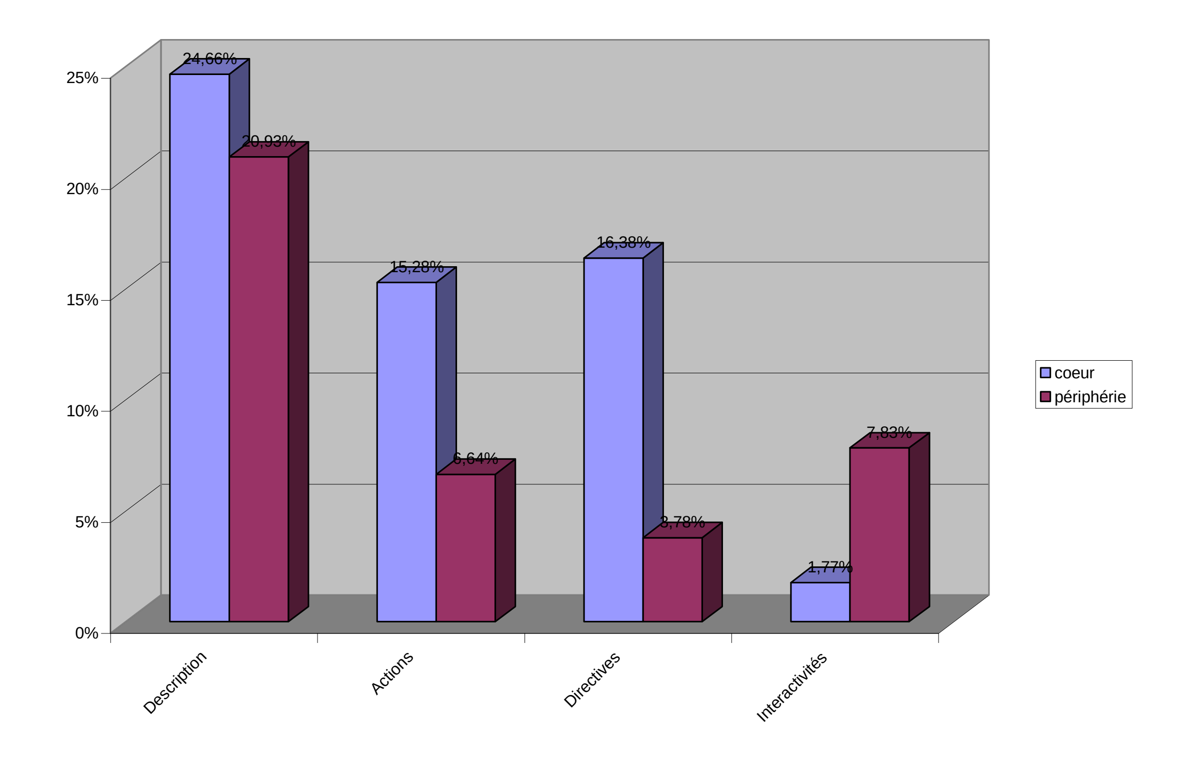
L'analyse de la spécificité de la catégorie description montre que dans une phase d'identification de déclaration du problème, la description permet aux participants de mettre en commun l'ensemble des informations concernant le problème et d'avoir par conséquent une représentation partagée du problème. Dans une phase de résolution, la description a pour but de mettre a disposition des informations concernant la solution et de construire collectivement la solution au problème.

A partir de la figure 6.10 et les résultats de l'analyse de spécificité , nous observons que dans les messages de la catégorie description émis par les développeurs du « core », les unités textuelles les plus spécifiques présentent des informations concernant une solution proposée ou une mise à jour d'une solution.

Dans les messages émis par les développeurs de la périphérie, la description est plus technique te concerne la description du problème plutôt que de la solution. Ces derniers contribuent à la résolution en fournissant des informations techniques permettant de reproduite le problème (url, #file) ou de l'identifier à un bug similaire (bug duplicate). Ils contribuent à la phase de construction de la solution avec des proportions moins importantes d'informations (suggestions).

En comparant les distributions des sous catégories « action » (cf. figure 6.11) dans les messages émis par les développeurs du « core » et de la « périphérie », nous constatons que les développeurs du « core » participent en majorité dans l'activité du développement par la proposition de solutions, la validation et la fixation du problème. Avec une forte proportion de la catégorie commande dans les messages émis par les développeurs du « core », nous remarquons que leurs actions s'accompagnent d'organisation. Comme l'indique la

figure 6.12, Les sous-catégories demande d'exécution d'action et instructions pour l'exécution d'action sont présent en majorité dans les commentaires émis par les développeurs du « core ».



Conclusion générale

Cette thèse a pour objectif l'étude de l'organisation des communautés Open Source dans le contexte d'une activité de résolution de problèmes distribuée. Les résultats auxquels ont abouti les analyses soulignent, d'une part, que la communication à travers le langage joue un rôle central dans l'organisation de la résolution de problèmes dans les contextes distribués. D'autre part, elles mettent en évidence les rôles des participants dans l'organisation de la résolution de problèmes en fonction de leurs positions hiérarchique dans la communauté. Dans ce sens, il s'avère que quelque soit le statut des participants, 'cœur' et 'périphérie' contribuent conjointement ou de manière solitaire à l'organisation de la résolution de problème, ils occupent néanmoins des rôles distincts, et impliquent des modes de coordination différents.

Avant de présenter les principaux apports, les limites et les perspectives, nous proposons une synthèse de la notre recherche.

I. Synthèse de la recherche

Cette recherche a été élaborée en deux phases : théorique et empirique. La phase théorique nous à permis de présenter une revue de la littérature sur la résolution de problèmes, sur l'organisation et la structuration de l'Open Source et sur la coordination dans les communautés Open Source. Cette revue a été menée dans les chapitre I, II et III.

La seconde phase de la recherche nous a permis de confronter les connaissances théoriques faites aux observations issues du terrain. Une démarche adductive a été adoptée (chapitre IV) consistant à faire un aller retour entre nos observations empiriques et la littérature pour définir nos questions de recherche.

Notre étude empirique a été conduite en trois phases : une phase exploratoire et deux phases constructives.

La phase exploratoire a consisté en une analyse d'un exemple de rapport de bogue. A travers une analyse qualitative inductive, cette phase nous a permis de dresser un diagnostic sur la résolution de problème au sein de Bugzilla et d'identifier différentes propriétés organisationnelles (chapitre V). Ensuite les résultats de la seconde phase de l'analyse ont permis, d'une part de vérifier les observations faites lors de la première phase exploratoire et de formuler d'autre part à travers une revue de la littérature, de nouvelles questions de

recherches. La méthode utilisée est linguistique qui allient à la fois des analyses de type qualitatives et quantitatives sur un corpus composé de 649 rapports de bogue.

Dans une troisième étape, nous avons mobilisé dans un premier temps, les mêmes méthodes de collecte et d'analyse utilisées dans la phase deux. Cette fois sur un échantillon plus important de 4 109 rapports de bogues critiques et avoir pour objectif l'identification des rôles des participants dans l'organisation de l'activité, en fonction de leurs statuts dans la communauté. La première série d'analyses a permis d'avoir une vue générale sur la contribution du cœur et de la périphérie. La deuxième série d'analyse est de type linguistique. La méthode d'analyse mobilisée est les spécificités, elle a permis de comparer les catégories linguistiques utilisées par les participants du cœur et de la périphérie, et d'identifier par la suite en fonction de la répartition de ces catégories leurs rôles dans l'organisation de l'activité au sein de Bugzilla.

Dans une deuxième étape, Nous avons mis en œuvre une analyse en composante principale pour identifier différents mode de coordinations en fonction de différentes propriétés organisationnelles identifiées dans les phases précédentes de l'analyse.

Au terme de cette thèse, il est important de revenir sur ses principaux apports avant d'en souligner les limites et d'en dégager les perspectives.

II . Les principaux apports de la recherche

Nous distinguons les apports théoriques, managériaux et méthodologiques.

II.A. Les apports théoriques

Notre recherche s'inscrit dans une démarche nouvelle dans la mesure où l'analyse de la communication entre les participants dans Bugzilla à été réalisée selon trois dimensions : une dimension organisationnelle, interactive et sociale. Ce travail confirme que l'organisation de la résolution de problèmes dans des contextes distribués se base sur la contribution du cœur et de la périphérie et que le rôle de ces derniers se distinguent et impliques des modes de coordinations différents

II.B. Les apports managériaux

Notre travail a permis de dresser un état des lieux aux entreprises qui souhaite adopter le modèle Open Source ou travaillant dans des contextes fortement distribuée. Les mécanismes de coordinations mis en place dans la cas Bugzilla peut les aider à prévoir la manière de gérer leur projets lorsqu'ils sont opérés à distance.

II.C. Apport méthodologique

L démarche empirique de ce travail de recherche offre l'originalité de pluralisme méthodologique. Une démarche qui se développe grandement en sciences de gestion. En effet, ce travail associe une approche quantitative à une approche qualitative. De plus l'approche méthodologique multidimensionnelle pour étudier l'organisation de l'activité dans ces communautés sur la base de trois dimensions : sociale, organisationnelle et interactive. Enfin, l'originalité de noter travail réside dans le fait que nous nous basons sur une méthode d'analyse linguistique, une méthode jusqu'as présent très peu explorer en science de gestion.

Références bibliographiques

Ayari, K., Meshkinfam, P., Antoniol, G., and Penta, M. D. (2007). Threats on building models from CVS and Bugzilla repositories: the Mozilla case study. In *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, pages 215–228, Richmond Hill, Ontario, Canada. ACM.

Bastien G. (2001), *Logiciel libre et innovation technique*, http://bastien1.free.fr/ILL.pdf

Baudry B. (1995), *L'économie des relations interentreprises*, Repères, La Découverte.

Bos, N., Olson G.M (2001), Being There versus Seeting There: Trust Via Video, *Working Paper, CREW*, Ann Arbor: University of Michigan, School of Information.

Bowles S. et Ginits H. (1998), How communities govern : the structural basic of prosocial norms, Economics, values and organisation, *Cambidge university Press*, Cambridge.

Cohendet P. et Diani M. (2003), l'organisation comme une communauté de communauté : croyance collectives et culture d'entreprise, *Revue d'Economie Politique*.

Coriat B. et Guennif S. (1996). Incertitude, confiance et institution, *Communication au Colloque 'La confiance en question'*, 22-23 mars, Aix-en-Provence.

Cowan R., P. David et D. Foray (2000), The Economics of Knowledge Codification andTacitness, *in Industrial and Corporate Change*, vol. 6, n° 3.

Cox, A. (1998). Cathedrals, bazaars and the town council. Available at http://www.linux.org.uk/Papers_CathPaper.cs.

Crowston, K. and Howison, J. (2005). The social structure of open source software development teams. *First Monday*, 10(2).

Crowston, K. and Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology & Policy*, 18(4):65–85.

Crowston, K., Heckman, R., Annabi, H., and Masango, C. (2005). A structuration perspective on leadership in free/libre open source software teams. In *Proceedings of the first international conference on open source systems*, pages 9–15. Genoa, Italy.

Crowston, K., Wei, K., Li, Q., and Howison, J. (2006). Core and periphery in free/libre open source software team communications. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICCS-39).*

Dalle J.-M. , Jullien N. (1999), « *Free Software under economic analysis : some insights from the economics of reativity* », présenté au colloque EMAEE, Gronoble, juin.

Dalle J-M. et Jullien N. (2003), « “Libre” software : turing fads into institutions ?”, *Resarch Policy, vol.32,* p.1-11.

Dalle, J. M.; David, P. A.; Ghosh, R. A. Et Wolak F. A. (2004), *Free and Source Software Developers and ‘the Economy of Regard’: Participation and Code-Sining in the Moduls of the Linux Kernel*, Aparticipation at OWLS:The Oxford Workshop on ‘Libre Source’ Convenend at the Oxford InternetInstitute,http://siepr.stanford.edu/programs/OpenSoftware_David/EconomyofRegard_8+_OWLS.pdf p 5.

Dalle, J.-M et Jullien, N. (2000), *Libre software : turning fads into institutions ?*, http:// opensource.mit.edu/papers/Libre-Software.

Dalle, J.-M. and David, P. (2007). Simulating code growth in libre (open-source) mode. In Curien, N. and Brousseau, E., editors, *The Economics of the Internet*. Cambridge University Press, Cambridge, UK.

Dalle, J.-M., den Besten, M., and Masmoudi, H. (2008). Channelling Firefox developers: Mom and dad aren’t happy yet. In *Proceedings of the Fourth International Conference on Open Source Systems*, Milan.

Dalle, J-M et David, P. A. (2003), The allocation of software development resources in open source production mode, *SIEPR Discussion Paper No. 02-27, Stanford University.*

Dasgupta, P. et **Paul, A. D.**(1987), Information Disclosure and the Economics of Science and Technology. ch. 16 in *Arrow and the Ascent of Modern Economic Theory*, (G. Feiwel, ed.), New York: New York University Press, 1987, pp. 519-542.

David, P. A., Waterman, A. et Arora, S. (2003), The Free/Libre/Open Source Software Survey for 2003, *Stanford institute for Economic Policy Research*.

den Besten, M., Dalle, J.-M., and Galia, F. (2008). The allocation of collaborative efforts in open-source software. *Information Economics and Policy*, 20(4):316–322.

Di Cosmo R. et Nora D. (1998), *Le Hold-up planétaire. La face cachée de Microsoft*, Calmann-lévy, 187p.

Fenneteau, H. et Guibert N. (1997**)**. *Trust in Buyer-Seller Relationship*: Toward a Dynamic Classification of the Antecedents, Document de recherche, I.A.E - Montpellier, CREGO, 2.

Ferrary M. et Vidal P. (2004), *Les leçons de management de la communauté Linux*, Conférence de l’AIMS, Le Havre.

Foray D. et Zimmermann J-B (2001), “l’économie du logiciel libre: organisation coopérative et incitation l’innovation”, *in Revue économique*, nº 52, p. 77-93.

Gensollen M. (1999). La création de valeur sur Internet, *Réseaux*, Vol. 17, n° 97, p. 15-76.
Gensollen M. (2003), « Biens informationnels et communautés médiatées », *revue d’économie politique.*

Ghosh R.A. , et David P.A. (2003), The Nature and Composition of the Linux Kernel Developer Community: a Dynamic Analysis, Working Paper, NFS Open Source Software Project, Stanford, *CA: Stanfoed Institute for Economic Policy Research.*

Harhoff H. et von Hippel (2003), Profiting from Voluntary Information Spillovers: how Users Benefit by Freely Revealing Their Innovations, *Research Policy*.

Himanen P. (2001), *The Hacker Ethic and the Spirit of the Information Age*. New York: Random House.

Holmström, B. 1999. Managerial Incentive Problems: A Dynamic Perspective. *Review of Economic Studies* 66 (169-182).

Jullien N. (2001), *Impact du logiciel sur l'industrie informatique*, thèse, http://www-eco.enst-bretagne.fr/Etudes projets/RNTL/documents universitaires.html.

Lakhani, K. R. et Wolf, R. (2003), *Why hackers do that they do : understanding motivations and effort in free/open source software projects*, http://opensource.mit.edu/papiers/lakhaniwolf.pdf.

Lakhani, K. R. et Wolf, R.G. (2003), *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*, MIT Sloan School of Management and The Boston Consulting Group.

Lakhani, K. R. (2006). *The Core and the Periphery in Distributed and Self-Organizing Innovation Systems*. PhD thesis, MIT.

Lanzara, G. F. and Morner, M. (2005). Artifacts rule! how organizing happens in open source software projects. In Czarniawska, B. and Hernes, T., editors, *Actor-Network Theory and Organizing*, pages 67–90. Copenhagen Business School Press.

Lave J. et E. C. Wenger (1991), Situated Learning: Legitimate Peripheral Participation, *Cambridge University Press New York, NY.*

Lebart, L. and Salem, A. (1994). *Statistique textuelle*.

Lebart, L., Salem, A., and Berry, L. (1998). *Exploring Textual Data*.

Lee, G. K. and Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization Science*, 14(6):633–649.

Lerner, J. et Tirole J. (2002), Some Simple Economics of Open Source. *Journal of Industrial Economics* , 197-234.

Lerner, J. et Tirole, J. (2002), "*The Simple Economics of Open Source*.", National Bureau of Economic Research (NBER) Working Paper 7600 (March), www.nber.org/papers/w7600.

Lerner, J. et Tirole, J. (2002), Some simple economics of open source, *The Journal of Industial Economics*, L(2), 197-234.

Lévy Pierre (1992), *de la programmation considérée comme des beaux-arts*, La Découverte, P245.

Lindenberg, S. (2001), *Intrinsic motivation in a new light*, Kyklos p.317-342.

Loilier T. et Tellier A. (2004), « Comment peut-on se faire confiance sans se voir ? Le cas du développement des logiciels libres », *M@n@gement, Vol. 7, No. 3, 275-306, Special Issue: Practicing Collaboration.*

Masmoudi, H., den Besten, M., de Loupy, C., and Dalle, J.-M. (2009). Peeling the onion: The words and actions that distinguish core from periphery in Firefox bug reports, and how they interact together. In Crowston, K. and Boldyreff, C., editors, *Proceedings of the Fifth International Conference on Open Source Systems*.

Maznevski, M. et Chudoba (2000), *"Bridging Space over Time : Global Virtual Team Dynamics and Effectiveness, Organisation Science"*, 473-492.

Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346.

Moon, J. Y. and Sproull, L. (2000). Essence of distributed work: The case of the Linux kernel. *First Monday*, 5(11).

Nakamura J. et Csikszentmihalyi. M. (2003), The construction of meaning through vital engagement. In *Flourishing: positive psychology and the life well-lived*, edited by C. L. Keyes and J. Haidt. Washington, DC: American Psychological Association.

Orléon, A. (1994), Sur le rôle respectif de la confiance et de l'intérêt dans la constitution de l'ordre marchand, *Cahier de recherche, CREA n° 9403A,* Paris : Ecole Polytechnique.

Ouchi W. (1979): *Markets, Bureaucracies and Clans*, Administrative Science Quarterly, 129-141.

Panteli, N. (2003), *Situating Trust Within Virtual Teams*, Working Paper, n°20, Bath: Uiversity of Bath, School of Managment.

Rayan, R. M. et Deci, E. L. (2000), Intrinsic and Extrinsic Motivations : Classic Definitions and New Directions, *Contemporary Education Psychology*.
Raymond E. S. (1998), *La Cathédrale et le bazard*, traduction de BLONDEEL Sébastien, http:// lifl.fr/~ blondeel/traduc/cathedral-bazaar/main_file.html.

Raymond E. S. (2000), *A la conquête de la noosphère, in Blondeau Olivier, Latrive Florent* (2000), Libres Enfants du savoir numérique, l'Eclat.

Raymond, E. S. (1999), "*A Response to Nikolai Bezroukov*, " *First Monday*, 4:11 (November 1999), http://firstmonday.org/issues/issue4_11/raymond/index.html .

Raymond, E. S. (1998). The cathedral and the bazaar. *First Monday*, 3.

Rossi C. et Bonaccorsi A., (2002), *Intrinsic vs. extrinsic incentives in profit-oriented firms supplying Open Source priducts and services*, http://www.firstmonday.org/issues/issue10_5/rossi/ .

Rousseau, D., Sitkin, S., Burt, R. et Camerer C. (1998), Not so different after all: A crossdiscipline view of trust, Academy of Management Review, Vol. 23, No. 3, p. 393-404.

Rullani, F. (2009). The periphery on stage: Functions, properties and dynamic evolution of the periphery in the free/open source software community. Unpublished report.

Ripoche, G. and Sansonnet, J.-P. (2006). Experiences in automating the analysis of linguistic interactions for the study of distributed collectives. *Computer Supported Cooperative Work*, 15:149–183.

Stallman R. M. (1998), “ Qu’est-ce que le logiciel libre? » in FSF and GNU.

Steinmuller E. (2002),“Virtual communities and the New Economy“ in Mansell R.(ed.): *Inside the communication Revolution, Oxford University Press, Oxford*.

Stewart D. (1984), *Secondary Research :Information Sources and Metuods*, Newbury Park, CA : Sage.

Tayon J. (2002), Le projet Linux est-il un modèle possible d’entreprise innovante ?.Cahier de recherche, CNAM, Chaire de développement des systèmes d’organisation.

Torvalds, L. et Diamond, D. (2001), Just for Fun : The Story of Accidental revolutionary, Texere.

Von Hippel, E. (2001), Innovation by user communities: Learning from open source software, *Sloan Managment Review.*

Villa, L. (2003). Large free software projects and bugzilla: Lessons from GNOME project QA. In *Proceedings of the Linux Symposium*, Ottawa, Canada.

Von Hippel, E. (2002), "Horizontal innovation networks - by and for users" Cambridge, MA, Massachussetts Institute of Technology, Sloan School of Management, *Working Paper No. 4366-02. June.*

Von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32:1217–1241.

Annexes

Annexes chapitre I

Annexe I.1Le processus de résolution de problème (Gasser et Sandusky, 2005)

Phase One: Problem Identification and Reporting
Step1. Problem Identification - User notices anomaly / mistake during normal use or testing - Actors identifying problems may be tested [1, 2][76] or customers / users [2] Encounter between a human and problematic phenomenon is subjective, complex, and does not necessarily result in the creation of a bug report Step2. Problem Reporting - Bug report created, enabling collective management of the SWPM Process - Actors creating the bug report may be tested [1] or intermediary (e.g., help desk) [2] Bug reports may represent enhancement request, accidental submissions, etc.
Phase Two: Bug Report Triage
- Problem re-creation is attempted - Priority and bug report assignment are assessed by a group of software designers whose primary role is the management of software problems [1] or by "marketing engineers" and assignment by "programming managers." [2] Reducing the number of bug reports needing attention (e.g., identifying duplicate bug reports) is a priority in the Mozilla community. Identification of the small number of high priority problems is also important.
Phase Three : Expert Analysis, Fixing, Testing, and Deployment
- Assignee (software designers [1] or software engineers [2]) investigates, determines cause, evaluates repair options, consults with other experts, coordinates the work of multiple experts, etc. - Assignee modifies and testes the software to resolve the bug and deploys the change if successful. Collective debugging, individual debugging, voting, and negotiation are some of the sensemarking, community-specific, and basic social processes employed by the Mozilla community in this phase of the SWPM process.
Phase Four: Bug Report Verification and Closure
Step1. Fix Verification - Quality assurance verifies that the bug has been corrected (the platform master responsible for the soft ware build [1] or the integration team [2]. - In the Mozilla community, the terminal status of most bug report is "verified". Step 2. Problem Closure - Bug report is marked "closed" by the central file manager [1] or the software engineer [2].

[76] Gasser et Sandusky (2005) font référence aux travaux de :

- Carstensen, P. H., Sørensen, C., Tuikka, T. (1995). Let's talk about bugs! *Scandinavian Journal of Information Systems*, 7(1), 33-54,
- Et Crowston, K. (1997). A coordination theory approach to organizational process design. *Organization Science*, 8(2),157-175.

Annexe I.2: Coded tasks in the bug fixing process Crowston et Scozzi (2004)

1.0.0 Submit (S) 1.1.0 Submit bug (code errors) 1.1.1 Submit symptoms 1.1.2 Provide code back trace (BT) 1.2.0 Submit problems 1.2.1 Submit incompatibility problems (NC)	**4.0.0 Fix** 4.1.0 Propose temporary solutions (AC) 4.2.0 Provide problem solution (SP) 4.3.0 Provide debugging code (F)
2.0.0. Assign 2.1.0 Bug self-assignment (A*) 2.2.0 Bug assignment (A)	**5.0.0 Test & Post** **5.1.0 Test/approve bug solution** 5.1.1 Verify application correctness W 5.2.0 Post patches (PP) 5.3.0 Identify further problems with pro-posted patch (FNW)
3.0.0 Analyze 3.1.0 Contribute to bug identification 3.1.1 Report similar problems ® 3.1.2 Share opinions about the bug (T) 3.2.0 Verify impossibility to fix the bug 3.2.1 Verify bug already fixed (AF) 3.2.2 Verify bug irreproducibility (NR) 3.2.3 Verify need for a not yet supported function(NS) 3.2.4 Verify identified bug as intentionally introduced (NCP) 3.3.0 Ask for more details 3.3.1 Ask for Code version/command line(V) 3.3.2 Ask for code back trace/examples (RBT/E) 3.4.0 Identify bug causes (G) 3.4.1 Identify and explain error (EE) 3.4.2 Identify and explain bug causes different from code(PNC)	**6.0.0 Close** 6.1.0 Close fixed bug/problem 6.2.0 Closed not fixed bug/problems 6.2.1 Close irreproducible bug (CNR) and close it 6.2.2 Close bug that asks for not yet supported function (CNS) 6.2.3 Close bug identified as intentionally introduced (CNCP)

Annexe I.3 : mozilla.org Staff Members (mozilla.org, 2008)[77]

If you have a project management issue, you can email the entire staff via staff@mozilla.org.

Mitchell Baker, Chief Lizard Wrangler (mitchell@mozilla.org)

Mitchell is the general troubleshooter, spokesperson and policy arbitrator for mozilla.org. She works extensively with companies and projects using Mozilla. This involves explaining mozilla.org processes, listening to the needs of contributors, and seeking to integrate open source development techniques with the world of commercial software development.

Mitchell also spends a lot of time driving consensus as to how mozilla.org ought to manage the project, which suggests she may have a masochistic streak. She dreams of making the Mozilla project easier to understand.

Before joining mozilla.org staff, Mitchell was the attorney at Netscape responsible for all legal issues related to product development and intellectual property protection. During that time she wrote the Netscape and Mozilla Public Licenses.

Chris Blizzard (blizzard@mozilla.org)

Chris hacks on various parts of Mozilla. The straight Xlib port of Mozilla is mostly his fault. He also hacks on the gtk port when it really needs help and people ask really nicely. He dreams of things like adding WebDAV support and other fun network features. He also dreams about the directions the mozilla project could take and what it can accomplish.

Chris has been using Linux and open software since the 0.99 days. Starting as a user he self taught himself programming and now hacks on various projects when he has time. He's been working with Mozilla code since the source was released. In the past he's played roles as a sysadmin, web jockey, database programmer and project manager.

Asa Dotzler (asa@mozilla.org)

Asa is our community quality advocate extraordinaire. He joined mozilla.org to turbo-charge our quality programs and to further develop our volunteer QA and testing community. When he's not helping new contributors test Mozilla and report bugs he's working with drivers@mozilla.org to define requirements for and ship Mozilla milestones.

Brendan Eich (brendan@mozilla.org)

[77] http://www-archive.mozilla.org/about/staff

Brendan is responsible for architecture and the technical direction of Mozilla. He is charged with authorizing module owners, owning architectural issues of the source base and writing the roadmap that outlines the direction of the Mozilla project.

Brendan created JavaScript, did the work through Navigator 4.0, and helped carry it through international standardization. Before Netscape, he wrote operating system and network code for SGI; and at MicroUnity, wrote micro-kernel and DSP code, and did the first MIPS R4K port of gcc, the GNU C compiler.

Gervase Markham (gerv@mozilla.org)

Gerv works part-time for the Mozilla Foundation, and is the only staff member based outside the continental USA. He is responsible for the relicensing project and trademarks, has a strong interest in security and usability, and does most of his hacking (when he has time) on Bugzilla. His blog is *Hacking for Christ*.

Myk Melez (myk@mozilla.org)

Myk is a toolsmith and sometimes document writer. Besides hacking on Bugzilla and occasionally coaxing it back to life when the world gets to be too much for it, he stands ready to write new tools (like Doctor) as the need arises. He tries his best to balance daily work with grand visions for the future.

Before mozilla.org, Myk wrote ForumZilla, before which he worked at UC Santa Cruz as a Web applications developer, before which he spent copious amounts of time, money, and energy getting a degree in the practical field of Sociology, before which he taught himself programming (apparently a common theme around these parts) on his sporty 0.3Mhz Apple IIe in order to hack keyboard support into video games.

MOZILLA.ORG STAFF ASSOCIATES

Staff Associate is an experimental position we created in the fall of 2001. The role is for a person who

- Has demonstrated experience and/or aptitude for addressing the types of issues that come to staff attention. These issues tend to be balancing of perspectives, communication and problem resolution, and fitting together differing needs. These are quite different from being a primo hacker, and require a different set of interests. Often the two do not overlap.
- Has been involved in Mozilla in a range of activities (i.e., code contributions are precious, but alone would be unlikely to make an Associate position make sense)
- Has demonstrated to staff an ability to work in our consensus based organizational style.
- Fills a hole in the expertise/resources of staff members.
- Is interested in the direction of Mozilla technology and in the Mozilla project in general, separate from an interest in any particular product
- can make decisions for the benefit of the entire community, both by temperament and because his or her employer (if any) permits this

An Associate supports mozilla.org staff by providing input into staff discussions and decision-making process. An Associate may also investigate issues and provide background materials to staff. An Associate does not speak for mozilla.org, and does not have an "@mozilla.org" email address.

The Associate role might be a step in joining mozilla.org staff, or it might show that the Associate and staff members don't work that well together. We may also use this role for staff members who no longer have enough time available to remain staff members but whose input we seek whenever we can get it.

mozilla.org Staff Associate Members

Peter Bojanic (peter@bojanic.ca)

Peter is an engineering type with that elusive combination of development experience and business acumen. He promotes Mozilla technology and translates tech talk to execs then reads their minds and explains the business speak to engineers. Peter works for OEone Corporation which is developing the Penzilla operating environment based on Linux and Mozilla. He's their VP Software Development and resident Jedi.

Scott Collins (scc@mozilla.org)

Scott is a core engineer, technical evangelist and liaison for the Mozilla project. You may already be familiar with Scott's work on XPCOM, nsCOMPtr, Mozilla's Strings, or from his technical lectures (e.g., on C++ portability [slides, video] Mozilla Strings [slides, video], XPCOM [video], or at campuses, conferences, and users' group meetings across the US). If you want to hack on Mozilla and don't know where to start or how to get involved, or if you're looking for someone to speak to your group about Mozilla engineering or technical details, Scott is here to help.

Frank Hecker (hecker@mozilla.org)

Frank Hecker is mozilla.org's self-appointed policy wonk and amateur document editor; among other things, he helped create the Mozilla Relicensing FAQ document, the mozilla.org License Policy, the policy for handling Mozilla security bugs, and the Mozilla Crypto FAQ document. Frank was one of the people who successfully sold Netscape management on the advantages of releasing source code, and in the spirit of "service after the sale" he has been volunteering with mozilla.org ever since.

Marcia Knous (marcia@mozilla.org)

Marcia can't be easily categorized in any technical sense since she doesn't have much of a technical background and often flies low on the radar.

Her mozilla.org work involves dealing with a lot of administrative minutia, including handling all CVS accounts and troubleshooting when things go awry. In her pre-Mozilla days, Marcia worked in entertainment and law and still dabbles in freelance writing. These days she mostly dreams about treating the anxieties of the present with the wisdom of the ages.

Dan Mosedale (dmose@mozilla.org)

Dan has been working on pieces of Mozilla (LDAP support, LDAP typedown autocomplete, getting UNIX Quantify to play nice) as well as mozilla.org tools (Bugzilla, Despot, license and statistics scriptage) for a while.

Before joining mozilla.org, Dan was the systems administrator at Mosaic Communications Corp. From there, he moved on to IS Architecture work at Netscape, co-wrote the DNS server used by Netcenter to distribute its Website worldwide, and enhanced and maintained Netscape's mail-to-news gateway code.

Mike Shaver (shaver@mozilla.org)

As a founding member of mozilla.org, Mike has enjoyed a rare opportunity to inflict a wide variety of trials and errors on the Mozilla code and project. He is stronger for it, and hopes that Mozilla is as well.

Scheming diabolically from his fortress of solitude in Toronto, shaver meddles in matters ranging from platform architecture and implementation to licensing and organizational development. If you are short on opinions, he often has some to spare.

Seth Spitzer (sspitzer@mozilla.org)

Back from exile, Seth is currently working on Firefox. Before that, he worked on Thunderbird and the mailnews part of the Mozilla Suite.

Annexes chapitre III

Annexe III.1 : Typologie des dépendances et des mécanismes de coordination (Malone et Crowston, 1994, p. 93)[78]

Dependency	Examples of coordination processes for managing dependency
Shared resources	"First come/first serve", priority order, budgets, managerial decision, market-like bidding
Task assignments	(Same as for "Shared resources")
Producer/consumer relationships	
Prerequisite constraints	Notification, sequencing, tracking
Transfer	Inventory management (e.g., "just in time, "economic order quantity")
Usability	Standardization, ask users, participatory design
Design for manufacturability	Concurrent engineering
Simultaneity constraints	Scheduling, synchronization
Task/subtask	Goal selection, task decomposition

[78] La traduction du tableau 2.1 est présentée en annexe 1

Annexe III.2 : Typologie des dépendances et des mécanismes de coordination (Crowston, 2003, p. 105)[79]

Task uses resource 1. determine needs 2. identify resources • ads • prepared list • only one resource 3. collect information on resources • by bidding • manager knows 4. pick best 5. do assignment • mark resource in use 6. manage flow dependencies from acquiring resource to using resource **Task requires multiple resources simultaneously** 1. pre-assign resources to simplify coordination problem 2. manage dependency on the fly • avoid or detect and resolve deadlock • detect and resolve starvation **Sharing: Multiple tasks use the same resource** • ensure same version of sharable resources • destroy obsolete versions • copy master prior to use • check versions prior to use • detect and fix problems after the fact • schedule use of non-shareable but reusable resources 1. check for conflict before using and then mark the resource as in-use 2. manage flow of resource from one task to another • allocate non-reusable resources • divide the resource among the tasks • abandon one task • get more resources **Flow: One task uses a resource created by another** 1. usability (i.e., the right thing) • user adapts to resource as created • creator gets information from user to tailor resource • 3rd party sets standard, followed by both producer and consumer 2. prerequisite (i.e., at the right time) • producer produces first • follow plan • monitor usage • wait to be asked • standard reorder points • when out

[79] La traduction du tableau 2.1 est présentée en annexe 2

• just-in-time
• consumer waits until produced
• monitor
• be notified
3. accessibility (i.e., in the right place)
• physical goods
• truck
• information
• on paper
• verbally
• by computer

Common output: Multiple tasks create the same output

1. Detect common output
• database of known problems
2. Manage common outputs
• effects overlap or are the same
• eliminate one task (manage shared resource)
• merge tasks take advantage of synergy
• effects are incompatible
• abandon one
• don't try to achieve them at the same time

Composition of tasks

• choose tasks to achieve a given goal (a planning problem)

Composition of resources

• trace dependencies of between resources to determine if a coordination problem exists

Annexes chapitre V

Annexe V.1: Descriptif du rapport (BR-362919)

Bugzilla@Mozilla – Activity log for bug 362919			Interaction	Interpretation
When / What	**Removed**	**Added**		
2006-12-06 01:10:17 PDT			*Description* *From* *Tim Maks van den Broek* when i use the preview pane in build version 2 beta 1 (20061205) the pane is to wide. The button "not scam" / show images are off screen. (see screenshot)	**Description du problème** La description du problème (commentaire 0, 1 et 2) se caractérise par une diffusion importante d'informations : -La présentation du problème (commentaire 0) -Diffusion d'éléments sur le contexte de la résolution : attachements dans les commentaires 1 et 2.
2006-12-06 01:16:48 PDT			*------- Comment #1 From Tim Maks van den Broek 2006-12-06 01:16:48 PDT ----*	

			--- Created an attachment (id=247651) [details] screenshot 1	Dans cette étape il n y a pas d'interactions entre développeurs. Le seul contributeur qui participe à la description du problème est le « bug reporter ». On constate également qu'il n y a pas de changement du statut du bug.
2006-12-06 01:17:40 PDT			------- *Comment #2 From Tim Maks van den Broek* Created an attachment (id=247652) [details] screenshot 2	
2006-12-06 08:13:23PDT **CC**		nirmalya.sensarma@gmail.com	------- *Comment #3 From Nir* when you open TB in safe mode , does it give same result ? http://kb.mozillazine.org/Safe_mode	Proposition de méthode de résolution avec un échange d'éléments contextuels. → échanges d'informations techniques sur le contexte du problème. Pas de changement du statut du bug.
2006-12-06 10:41:24 PDT **Severity**	normal	trivial	------- *Comment #4 From Mike Cowperthwaite* In TB 2, the alert bar no longer wraps its text when it's made narrow; it has a minimum width	Re-description du problème Définition du statut du bug : - changement du niveau de « sévérité »
Keywords		regression		

OS/Version	Linux	All	which (in the English version, anyway) is wide enough for the text "To protect your privacy..." plus the icon and the Load Images button. If the window is narrowed below that width, the window edge will start to encroach on the bar (and on the message body, too, which never gets narrower than the alert bar, if there is one). It's true that, with the spam or scam alerts, there is now a lot of blank space in the alert bar at the minimum width. Please don't set a target when you're filing a bug; that field is for use by the developers.	(normal—trivial). Le message présentes des éléments techniques. - interactions sur des éléments techniques.
Platform	PC	All		
Target Milestone	Thunderbird2.0	---		
Version	unspecified	2.0		
2006-12-06 10:45:38 PDT **Summary**	preview pane is to big (scam/spam)	message pane is too wide due	mcow@well.com	

		to alert bar's minimum-width		
2006-12-06 11:11:46 PDT **Flag**		blocking-thunderbird2+	*Comment #5 From Scott MacGregor* Two things jump out at me here: 1) The "to protect your privacy" text should still wrap, I'm not sure why that broke, I could have sworn it worked when I first landed the UI changes to the privacy bar. 2) The text shouldn't have a minimum width. It's designed to take as much space as it needs, then there's a spring that keeps the button as far right as possible. But when I shrink the window now, it sure looks like something is keeping that from	- Interprétation du problème - Collaboration technique pour la définition du problème. - Diffusion de plus d'informations sur l'environnement du problème

			working right.	
2006-12-07 00:06:46 PST			Comment #6 Tim Maks van den Broek (In reply to comment #3) > when you open TB in safe mode , does it give same result ? > http://kb.mozillazine.org/Safe_mode yes if the folderpane is at the same width.	Réaction du bug reporter : échanges directs entre répondant pour la description du problème : - réponse a la question postée dans les commentaires 6 et 7(commentaire 6), → échanges d'informations contextuelles.
2006-12-07 00:13:34 PST			Comment #7 Tim Maks van den Broek (In reply to comment #4) > minimum width which (in the English version, anyway) is wide enough for the> text "To protect your privacy..." plus the icon and the Load Images button. that is very bad for l10n, in i.e. nl the text is much longer	

Bug duplicate :
Problèmes similaires
Persistance du problème

2006-12-12 10:52:11 PDT CC		reiferreira @click21. com.br	*Comment #8 From Mike Cowperthwaite 2006-12-12 10:52:11 PDT* ------- *** ~~Bug 363582~~ has been marked as a duplicate of this bug. ***	
2006-12-18 08:33:04 PDT CC		hendrik.m aryns@uni tuebingen. de	*Comment #9 From Magnus Melin* *** ~~Bug 364211~~ has been marked as a duplicate of this bug. ***	
2006-12-18 08:33:15 PDT CC		mkmelin+ mozilla@i ki.fi	mkmelin+mozilla@iki.fi	
2007-01-15 15:48:30 PDT **Attachment #251588**		review?(ne il@httl.net)	------- *Comment #10 From Scott MacGregor* ------- ~~Created an attachment (id=251588) [details]~~ agressive fix This patch fixes two problems:the containing vbox needed flex in order to allow the remote image description text to wrap.	**Phase de resolution** → **Patch 1** - Proposition de solution "patch" - Une description du patch « this paatch fixes two problems ». - demande de révision du patch « Neil for a

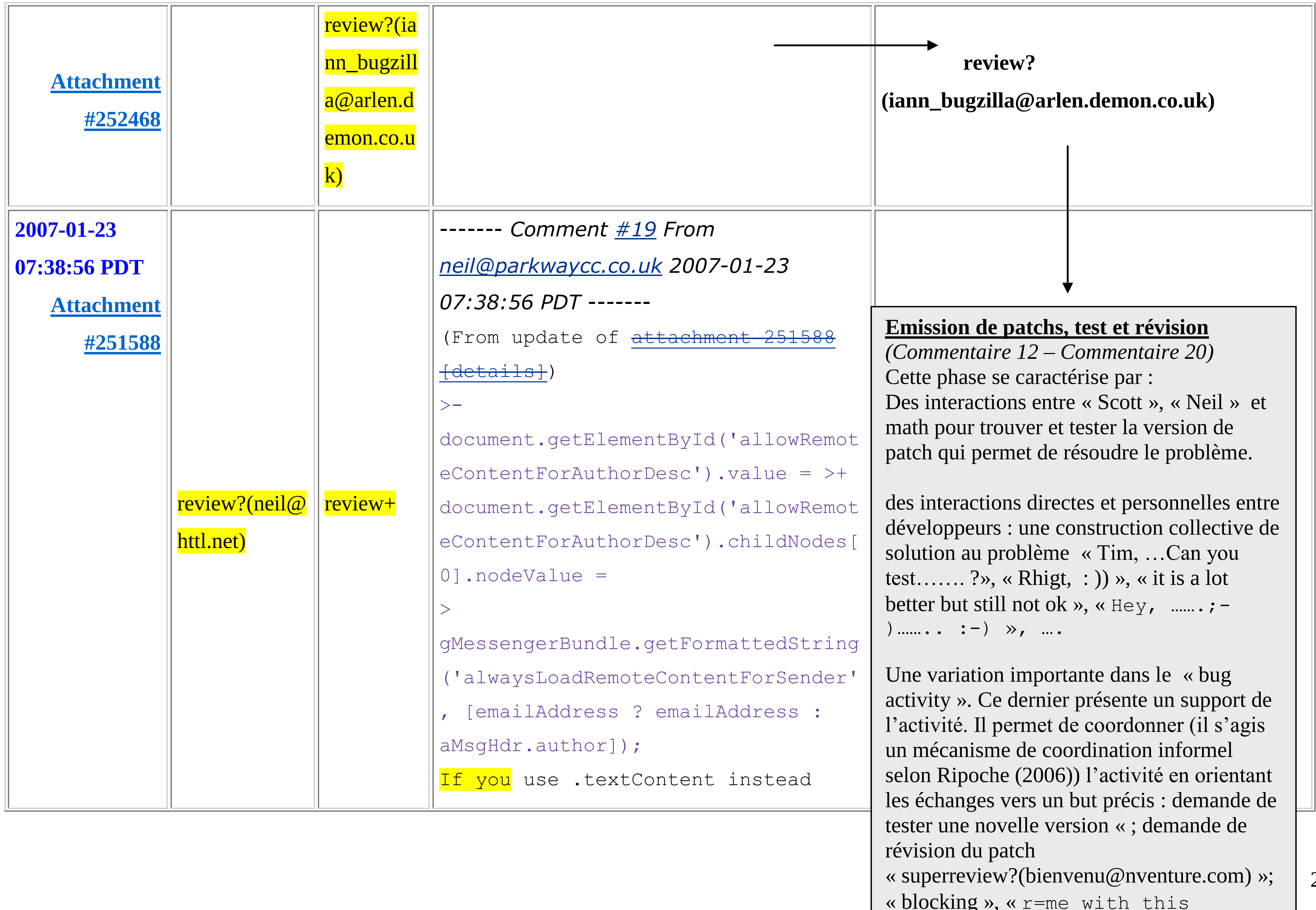
			In order to make things work exactly like they did in 1.5.0.9, I had to also revert the label back to a description element but as Neil pointed out the text-link class is really for labels not descriptions. Not sure which trade off to make here; One comprise might be to just add the flex to the box, this fixes the initial junk bar problem reported here. But would be nice to get that 2nd line of text for the remote image bar to also wrap. Partially asking Neil for a review/ advice on what to do with the label.	review(review?(neil@httl.net) ». Dans ce cas le répondant s'adresse directement à Neil pour réviser le patch. → Demande d'exécution d'action
2007-01-15 21:07:39 PDT [**Attachment #251601**]		superreview?(bienvenu@nvent	------- *Comment #11 From Scott MacGregor* ~~Created an attachment (id=251601) [details]~~ lower risk, complementary patch	→ **Patch 2** Nouveau « patch » : amélioration de la dernière

		ure.com)	this makes things a bit better as the description text will start wrapping again, but the dynamically generated label still won't wrap until Neil and I figure something out. It's still worth adding the flex to the box as it does make this problem better.	version. Collaboration entre Neil et Scott pour la proposition de solution. Demande de révision du patch « superreview?(bienvenu@nventure.com) »
2007-01-16 08:01:52 PDT **Attachment #251601**	superreview? (bienvenu@nventure.com)	superreview+	bienvenu@nventure.com	Acceptation de la révision du patch : Test du patch de la part du « bug reporter ». superreview+ (bienvenu@nventure.com)
2007-01-16 04:43:53 PDT			*Comment #12 From neil@parkwaycc.co.uk* (In reply to comment #10) >In order to make things work exactly like they did in 1.5.0.9, I had to also revert	

			>the label back to a description element but as Neil pointed out the text-link class >is really for labels not descriptions. Not sure which trade off to make here. One option is to add the crop attribute to the label. I tried <label class="text-link">foo</label> on trunk but it didn't underline.	
2007-01-16 13:14:56 PDT			------- *Comment #13 From Scott MacGregor* Tim, this should be a lot better in beta 2. I landed the "lower risk, complementary fix" in time for beta 2 last night. Can you test today's nightly or beta 2 when it comes out?	
2007-01-17 11:53:22 PDT			*Comment #14 From Tim Maks van den Broek* Created an attachment (id=251811)	

			[details] screenshot 3 it is a lot better but still not ok. see this screenshot	Test 1 du patch 2
2007-01-17 20:33:44 PDT			---- *Comment #15 From Scott MacGregor* right, the problem with the remote content bar requires a fix like the more aggressive patch (or at least a version that works better than that one :)). The simple fix that went in addressed the problem for the junk and phishing bar.	
2007-01-17 23:57:31 PDT			------- *Comment #16 From Tim Maks van den Broek* Created an attachment (id=251884) [details] screenshot 4 it didn't fix that problem also not totally	
2007-01-21 14:12:50 PDT Attachment #252252		review?(ia nn_bugzill	------- *Comment #17 From neil@parkwaycc.co.uk* ~~Created an attachment (id=252252) [details]~~	

		a@arlen.demon.co.uk)	Suite version While I was there I thought I'd clean up some of the other flexes too. I also added the crop="end" on the label as an interim fix. I think the lack of underlining on <label class="text-link">Click here...<label> might be a Gecko bug.	
2007-01-23 07:30:46 PDT [Attachment #252252]	review?(iann_bugzilla@arlen.demon.co.uk)		------- *Comment #18 From neil@parkwaycc.co.uk* Created an attachment (id=252468) [details] Fixed suite patch Filed the underlining issue on trunk as bug 367745.	
Attachment #252252 is **obsolete**	0	1		



Attachment #252468		review?(iann_bugzilla@arlen.demon.co.uk)		**review? (iann_bugzilla@arlen.demon.co.uk)**
2007-01-23 07:38:56 PDT Attachment #251588	review?(neil@httl.net)	review+	------- *Comment #19 From neil@parkwaycc.co.uk 2007-01-23 07:38:56 PDT -------* (From update of ~~attachment 251588~~ ~~[details]~~) >- document.getElementById('allowRemoteContentForAuthorDesc').value = >+ document.getElementById('allowRemoteContentForAuthorDesc').childNodes[0].nodeValue = > gMessengerBundle.getFormattedString('alwaysLoadRemoteContentForSender', [emailAddress ? emailAddress : aMsgHdr.author]); If you use .textContent instead	

<u>Emission de patchs, test et révision</u>
(Commentaire 12 – Commentaire 20)
Cette phase se caractérise par :
Des interactions entre « Scott », « Neil » et math pour trouver et tester la version de patch qui permet de résoudre le problème.

des interactions directes et personnelles entre développeurs : une construction collective de solution au problème « Tim, …Can you test……. ?», « Rhigt, :)) », « it is a lot better but still not ok », « Hey, …….;-)……. :-) », ….

Une variation importante dans le « bug activity ». Ce dernier présente un support de l'activité. Il permet de coordonner (il s'agis un mécanisme de coordination informel selon Ripoche (2006)) l'activité en orientant les échanges vers un but précis : demande de tester une novelle version « ; demande de révision du patch « superreview?(bienvenu@nventure.com) »; « blocking », « r=me with this fixed »;

			(see attachment 252468 [details]) you don't need the foo. >-<label id="allowRemoteContentForAuthorDesc" class="text-link" flex="1">- onclick="allowRemoteContentForSender();"/> >+<description id="allowRemoteContentForAuthorDesc" class="text-link" flex="1" >+ onclick="allowRemoteContentForSender();">foo</description> Leave this as a label. r=me with this fixed.	
	review?(iann_		------- *Comment #20 From Ian Neal* (From update of attachment 252468 [details])	

2007-01-23 14:45:14 PDT **Attachment #252468**	bugzilla@arlen.demon.co.u)	review+	r=me definite improvement →	**review+ iann_bugzilla@arlen.demon.co.uk**
2007-01-23 17:58:58 PDT **Attachment #251588 is obsolete**	0	1	*Comment #21 From Scott MacGregor* Created an attachment (id=252553) [details] updated tbird patch → updated patch based on Neil's review comments and to pick up the flex changes he made to the suite which look good too.	**Patch 3**
Attachment #251601 is obsolete	0	1		
Attachment #252553		superreview+		**Fixation du patch 3**
2007-01-23 18:01:18 PDT	NEW	RESOLVED	-- *Comment #22 From Scott MacGregor* Neil, do you want to nominate the	

Status			suite patch for the branch?	
Keywords		fixed1.8.1.2		
Resolution		FIXED		

Bugzilla@Mozilla – Bug 362919	message pane is too wide due to alert bar's minimum-width	Last modified: 2007-08-21 12:10:57 PDT

Summary: message pane is too wide due to alert bar's minimum-width

Status:	RESOLVED FIXED	**Reported:**	2006-12-06 01:10 PST by Tim Maks van den Broek
Whiteboard:		**Modified:**	2007-08-21 12:10 PDT (History)
Keywords:	fixed-seamonkey1.1.1, fixed1.8.1.2, regression	**CC List:**	8 users (show)
Product:	Thunderbird		
Component:	Mail Window Front End	**Flags:**	
Version:	2.0	mscott: blocking-thunderbird2+	
Platform:	All All	**See Also:**	
Importance:	-- trivial (vote)	**Crash Signature:**	
Target Milestone:	---		
Assigned To:	Scott MacGregor		
QA Contact:	front-end		
URL:			
Duplicates:	341998 363582 364211 369187 371185 (view as bug list)		
Depends on:			
Blocks:			
	Show dependency tree / graph		

Description **Tim Maks van den Broek 2006-12-06 01:10:17 PST**
when i use the preview pane in build version 2 beta 1 (20061205) the pane is to wide. The button "not scam" / show images are off screen. (see screenshot)

Comment 1 **Tim Maks van den Broek 2006-12-06 01:16:48 PST**
Created attachment 247651 [details]
screenshot 1

Comment 2 **Tim Maks van den Broek 2006-12-06 01:17:40 PST**
Created attachment 247652 [details]
screenshot 2

Comment 3 **Nir 2006-12-06 08:13:23 PST**
when you open TB in safe mode , does it give same result ?
http://kb.mozillazine.org/Safe_mode

Comment 4 **Mike Cowperthwaite 2006-12-06 10:41:24 PST**
In TB 2, the alert bar no longer wraps its text when it's made narrow; it has a minimum width which (in the English version, anyway) is wide enough for the text "To protect your privacy..." plus the icon and the Load Images button. If the window is narrowed below that width, the window edge will start to encroach on the bar (and on the message body, too, which never gets narrower than the alert bar, if there is one).

It's true that, with the spam or scam alerts, there is now a lot of blank space in the alert bar at the minimum width.

Please don't set a target when you're filing a bug; that field is for use by the developers.

Comment 5 **Scott MacGregor 2006-12-06 11:11:46 PST**
Two things jump out at me here:

1) The "to protect your privacy" text should still wrap, I'm not sure why that broke, I could have sworn it worked when I first landed the UI changes to the privacy bar.

2) The text shouldn't have a minimum width. It's designed to take as much space as it needs, then there's a spring that keeps the button as far right as possible. But when I shrink the window now, it sure looks like something is keeping that from working right.

Comment 6 **Tim Maks van den Broek 2006-12-07 00:06:46 PST**
(In reply to comment #3)
> when you open TB in safe mode , does it give same result ?
> http://kb.mozillazine.org/Safe_mode
>

yes if the folderpane is at the same width.

Comment 7 **Tim Maks van den Broek 2006-12-07 00:13:34 PST**

(In reply to comment #4)
>
> minimum width which (in the English version, anyway) is wide enough for the
> text "To protect your privacy..." plus the icon and the Load Images button.

that is very bad for l10n, in i.e. nl the text is much longer.

Comment 8 **Mike Cowperthwaite 2006-12-12 10:52:11 PST**

*** Bug 363582 has been marked as a duplicate of this bug. ***

Comment 9 **Magnus Melin 2006-12-18 08:33:04 PST**

*** Bug 364211 has been marked as a duplicate of this bug. ***

Comment 10 **Scott MacGregor 2007-01-15 15:48:30 PST**

Created attachment 251588—[diff]—[details] [review]
agressive fix

This patch fixes two problems:

the containing vbox needed flex in order to allow the remote image description text to wrap.

In order to make things work exactly like they did in 1.5.0.9, I had to also revert the label back to a description element but as Neil pointed out the text-link class is really for labels not descriptions. Not sure which trade off to make here.

One comprise might be to just add the flex to the box, this fixes the initial junk bar problem reported here. But would be nice to get that 2nd line of text for the remote image bar to also wrap.

Partially asking Neil for a review / advice on what to do with the label.

Comment 11 **Scott MacGregor 2007-01-15 21:07:39 PST**

Created attachment 251601—[diff]—[details] [review]
lower risk, complementary patch

this makes things a bit better as the description text will start wrapping again, but the dynamically generated label still won't wrap until Neil and I figure something out. It's still worth adding the flex to the box as it does make tis problem better.

Comment 12 **neil@parkwaycc.co.uk 2007-01-16 04:43:53 PST**

(In reply to comment #10)
>In order to make things work exactly like they did in 1.5.0.9, I had to also revert
>the label back to a description element but as Neil pointed out the text-link class
>is really for labels not descriptions. Not sure which trade off to make here.

One option is to add the crop attribute to the label.

I tried <label class="text-link">foo</label> on trunk but it didn't underline.

Comment 13 Scott MacGregor 2007-01-16 13:14:56 PST

Tim, this should be a lot better in beta 2. I landed the "lower risk, complementary fix" in time for beta 2 last night. Can you test today's nightly or beta 2 when it comes out?

Comment 14 Tim Maks van den Broek 2007-01-17 11:53:22 PST

Created attachment 251811 [details]
screenshot 3

it is a lot better but still not ok.
see this screenshot

Comment 15 Scott MacGregor 2007-01-17 20:33:44 PST

right, the problem with the remote content bar requires a fix like the more aggressive patch (or at least a version that works better than that one :)). The simple fix that went in addressed the problem for the junk and phishing bar.

Comment 16 Tim Maks van den Broek 2007-01-17 23:57:31 PST

Created attachment 251884 [details]
screenshot 4

it didn't fix that problem also not totally

Comment 17 neil@parkwaycc.co.uk 2007-01-21 14:12:50 PST

Created attachment 252252–[diff]–[details] [review]
Suite version

While I was there I thought I'd clean up some of the other flexes too.
I also added the crop="end" on the label as an interim fix.
I think the lack of underlining on <label class="text-link">Click here...<label> might be a Gecko bug.

Comment 18 neil@parkwaycc.co.uk 2007-01-23 07:30:46 PST

Created attachment 252468 [diff] [details] [review]
Fixed suite patch

Filed the underlining issue on trunk as bug 367745.

Comment 19 neil@parkwaycc.co.uk 2007-01-23 07:38:56 PST

Comment on attachment 251588–[diff]–[details] [review]
agressive fix

```
>-    document.getElementById('allowRemoteContentForAuthorDesc').value =
```

```
>+    document.getElementById('allowRemoteContentForAuthorDesc').childNodes[0].nodeValue =
>       gMessengerBundle.getFormattedString('alwaysLoadRemoteContentForSender', [emailAddress ? emailAddress :
aMsgHdr.author]);
```

If you use .textContent instead (see attachment 252468 [diff] [details] [review]) you don't need the foo.

```
>-        <label id="allowRemoteContentForAuthorDesc" class="text-link" flex="1"
>-               onclick="allowRemoteContentForSender();"/>
>+        <description id="allowRemoteContentForAuthorDesc" class="text-link" flex="1"
>+               onclick="allowRemoteContentForSender();">foo</description>
```

Leave this as a label. r=me with this fixed.

Comment 20 Ian Neal 2007-01-23 14:45:14 PST

Comment on attachment 252468 [diff] [details] [review]
Fixed suite patch

r=me definite improvement

Comment 21 Scott MacGregor 2007-01-23 17:58:58 PST

Created attachment 252553 [diff] [details] [review]
updated tbird patch

updated patch based on Neil's review comments and to pick up the flex changes he made to the suite which look good too.

Comment 22 Scott MacGregor 2007-01-23 18:01:18 PST

Neil, do you want to nominate the suite patch for the branch?

Comment 23 neil@parkwaycc.co.uk 2007-01-24 05:27:06 PST

(In reply to comment #22)
>Neil, do you want to nominate the suite patch for the branch?
Hey, let me check it in on trunk first ;-)

As this is suite-only code I'd normally set our approval-seamonkey1.1.1 flag but as this product doesn't have it I'll just poke someone on IRC :-)

Comment 24 Robert Kaiser (:kairo@mozilla.com) 2007-01-24 06:18:15 PST

approval-seamonkey1.1.1=me

Comment 25 Giacomo Magnini 2007-01-24 06:51:54 PST

*** Bug 341998 has been marked as a duplicate of this bug. ***

Comment 26 Magnus Melin 2007-02-04 03:31:40 PST

*** Bug 369187 has been marked as a duplicate of this bug. ***

Comment 27 Robert Kaiser (:kairo@mozilla.com) 2007-03-02 05:48:05 PST

*** Bug 371185 has been marked as a duplicate of this bug. ***

Comment 28 **Scott MacGregor 2007-08-21 12:10:57 PDT**

```
I think there was an unintended side effect with using .textContent for the text-link label in ths patch. The label
is no longer underlined like it should be since it's a link. Neil, do you know why using .textContent would have
broken that?
```

Annexe V. 3: Activité du bogue (BR-362919)

Bugzilla@Mozilla – Activity log for bug 362919: message pane is too wide due to alert bar's minimum-width

Who	When	What	Removed	Added
nir.sen	2006-12-06 08:13:23 PST	CC		nirmalya.sensarma
mcow	2006-12-06 10:41:24 PST	Severity	normal	trivial
		Keywords		regression
		OS	Linux	All
		Hardware	PC	All
		Target Milestone	Thunderbird2.0	---
		Version	unspecified	2.0
mcow	2006-12-06 10:45:38	Summary	preview pane is to big (scam/spam)	message pane is too wide due to alert bar's minimum-width

	PST			
mscott	2006-12-06 11:11:46 PST	Flags		blocking-thunderbird2+
mcow	2006-12-12 10:52:11 PST	CC		reiferreira
mkmelin+mozilla	2006-12-18 08:33:04 PST	CC		hendrik.maryns
mkmelin+mozilla	2006-12-18 08:33:15 PST	CC		mkmelin+mozilla
mscott	2007-01-15 15:48:30 PST	Attachment #251588 Flags		review?(neil)
mscott	2007-01-15 21:07:39 PST	Attachment #251601 Flags		superreview?(bienvenu)
dbienvenu	2007-01-16 08:01:52 PST	Attachment #251601 Flags	superreview?(bienvenu)	superreview+
neil	2007-01-21 14:12:50 PST	Attachment #252252 Flags		review?(iann_bugzilla)

neil	2007-01-23 07:30:46 PST	Attachment #252252 Flags	review?(iann_bugzilla)	
		Attachment #252252 Attachment is obsolete	0	1
		Attachment #252468 Flags		review?(iann_bugzilla)
neil	2007-01-23 07:38:56 PST	Attachment #251588 Flags	review?(neil)	review+
iann_bugzilla	2007-01-23 14:45:14 PST	Attachment #252468 Flags	review?(iann_bugzilla)	review+
mscott	2007-01-23 17:58:58 PST	Attachment #251588 Attachment is obsolete	0	1
		Attachment #251601 Attachment is obsolete	0	1
		Attachment #252553 Flags		superreview+
mscott	2007-01-23 18:01:18 PST	Status	NEW	RESOLVED
		Keywords		fixed1.8.1.2
		Resolution	---	FIXED

kairo	2007-01-24 06:18:15 PST	CC		kairo
prometeo.bugs	2007-01-24 06:51:54 PST	CC		prometeo.bugs
neil	2007-01-24 10:01:13 PST	Keywords		fixed-seamonkey1.1.1
mkmelin+mozilla	2007-02-04 03:31:40 PST	CC		oooolivier
kairo	2007-03-02 05:48:05 PST	CC		gweismann

Index des figures

Index des tableaux

Résumé:

Le contexte de cette thèse est l'étude de l'organisation de l'Open Source. L'objectif est d'étudier des phénomènes émergeant d'une communauté rassemblée autour de la résolution de bogues dont les principales caractéristiques sont la forte distribution de l'activité et des participants.

Nous considérons dans ce travail que la communication à travers le langage joue un rôle central dans l'organisation de la résolution de problèmes dans des contextes distribués. A travers l'étude du cas de Bugzilla, nous montrons à travers une première phase d'analyse linguistique que la communication englobe à la fois les activités liées à la définition du problème et à l'élaboration de la solution et les activités interactionnelles. L'analyse linguistique nous a également permis de comprendre les rôles des participants dans l'organisation de la résolution de problèmes en fonction de leurs positions hiérarchique dans la communauté.

Enfin, une analyse en composante principale sur les différentes propriétés organisationnelles, déterminées dans les premières phases la recherche, a mis en évidence différents modes de coordination.

Cette recherche propose une nouvelle vision de l'organisation de la résolution de problèmes dans des contextes fortement distribués. Elle montre que quelque soit le statut des participants, 'cœur' et 'périphérie' contribuent conjointement ou de manière isolée à l'organisation de la résolution de problèmes. Ils occupent néanmoins des rôles distincts, et se coordonnent selon des modes différents.

Mots clés : Communauté Open Source, Résolution de Problèmes, Distribution, Organisation, Communication, Langage.

Abstract:

The context of this research is the study of Open Source's organization. The aim is to study phenomena of a community gathered around the resolution of bugs, which the main characteristics are the strong (important) distribution of the activity and the participants.

We consider in this work that the communication through the language plays a central role in the organization of problems solving in distributed contexts. Through the study of the case of Bugzilla, we show through a first phase of linguistic analysis that the communication includes at once the activities bound (connected) to the definition of the problem and to the elaboration of the solution and the interactional activities. The linguistic analysis also allowed us to understand the roles of the participants in the organization of the problems solving according to their hierarchical positions in the community.

Finally, an analysis in main constituent principal component analysis on the various organizational properties, determined in the first phases of this research, brought to light various modes of coordination.

This research proposes a new vision of the organization of problems solving in strongly distributed contexts. It shows that the participants, 'core' and 'periphery' contribute collectively or in an isolated way to the organization of the problems solving. They occupy nevertheless different roles, and coordinate according to different modes.

Keywords: Open Source community, Distribution, Organization, Communication, language.